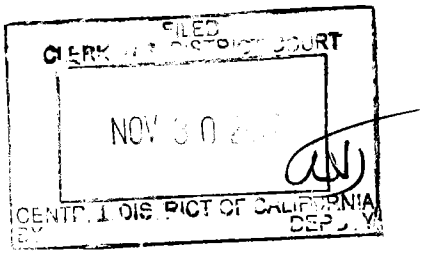
Louis A. Coffelt, Jr.
email: Louis.Coffelt@gmail.com
231 E. Alessandro Blvd., Ste 6A-504
Riverside, CA 92508
Phone: (951) 790-6086
Pro Se



UNITED STATES DISTRICT COURT

for the Central District of California

Louis A. Coffelt, Jr.,
Plaintiff,
--v.--
Autodesk, Inc.,
Defendant.

5:17-CV-01684-FMO-SHK

FIRST AMENDED COMPLAINT FOR COPYRIGHT INFRINGEMENT

JURY TRIAL DEMAND

ORIGINAL

JURISDICTION

1. This Court has subject matter jurisdiction pursuant to 17 U.S.C. §§ 101, et. seq., and 28 U.S.C. §§ 1331 and 1338(a) any Act of Congress relating to patents, copyrights, and trademarks.

2. This Court has personal jurisdiction over Defendant Autodesk, Inc. based on the allegation that Defendant committed and continues to commit acts of infringement in violation of 17 U.S.C. §§ 101, et. seq., and 17 U.S.C. § 501(a). Furthermore, based on the allegation that Autodesk, Inc. places infringing products into the stream of commerce, and Defendant has the knowledge or understanding that such products are sold in the State of California, including this Central District of California. Based on information and belief, Autodesk, Inc. has substantial revenue from the sale of infringing products within this District, expect their actions to have consequences in this District, and derive substantial revenue from the infringing products through interstate and international commerce.

VENUE

3. Venue is proper within this District under 28 U.S.C. § 1391(b),(c) based on the allegation that Autodesk, Inc., transacts business in this District, and offers for sale in this District products which infringe Plaintiff's copyrights. Furthermore, venue is proper in this District based on the fact that Plaintiff resides in this District, and Plaintiff incurred injuries in this District. Pursuant to Local Rule 3-2(c), Intellectual Property Actions are assigned on a district-wide basis.

PARTIES

4. Plaintiff's name is Louis A. Coffelt, Jr. referred to herein as (Coffelt). Coffelt resides at 5300 Herrera Ct., Riverside, CA 92505.

5. A first Defendant is Autodesk, Inc. referred to herein as (Autodesk), having a Corporate office at 111 McInnis Parkway, San Rafael, CA 94903.

INTRODUCTION

6. Plaintiff, Coffelt is the author of Photorealistic computer aided design (CAD). Digital images now have the appearance of a photograph of real objects (photorealistic). For example, On August 21, while Coffelt is filing a document with the District Court Clerk, there is a total solar eclipse occurring in Piedmont Missouri, Silver Lake Missouri, St Louis Missouri, Farmington Missouri, and Perryville Missouri; and Coffelt's copyrighted work will derive a concise digital image of the corresponding shadow for any specific resolution implemented.

There are 3 distinct programs directed to Coffelt's Photorealistic results:

(a) Vector Plane Intersection;

(b) Surface Shading by Reflective Intensity;

(c) Steradian Space for Light Occlusion Derivation.

Coffelt is the sole owner of all rights title and interest in Coffelt's programs. United States Certificates of Registration have been issued for Coffelt's Literary Works.

7. Coffelt applied more than 10,000 hours of work directed to development of Coffelt's CAD programs and support programs. These 10,000 hours of Coffelt's work occurred between the year 2010 through 2014. Coffelt created more than 50,000 digital files related to Coffelt's copyrighted works.

8. Photorealistic CAD programs do not exist prior to Coffelt's copyrighted works. Photorealistic CAD images do not exist prior to Coffelt's copyrighted works.

9. Starting about the year 1970 through about 2010, ("ray tracing") is the foundation of CAD. More than 200 lines of source code is iterated millions of times in order to derive one pixel in a bitmap. For example, millions of rays are cast into a CAD scene, where only a few thousand rays will create a graphic object. Ray tracing is essentially a method to search for graphic objects. Ray tracing is well-known to be inaccurate.

10. Starting about the year 1970 through about 2010, all graphic surfaces in CAD are polygon approximations. For example, a specific set of flat polygons are used to approximate a spherical surface. Ray tracing is used to find an intersection with each polygon in order to create the image of the sphere. Realistic smooth curved surfaces do not exist in this 40 year period.

11. Starting about the year 1970 through about 2010, non-realistic surface shading is the state of the art for CAD. All surfaces in this period are polygon approximations. Surfaces are not realistic with polygon approximations. Therefore, realistic surface shading can not exist in the domain of polygon approximations.

12. Starting about the year 1970 through about 2010, 2 dimensional shadow maps is the state of art. For more than 40 years, CAD programs create only 2 dimensional shadows. For more than 40 years, 2 dimensional shadows is the expected result.

13. For more than 40 years, CAD programs required millions of CPU clock cycles in order to derive one pixel in a bitmap. In comparison, Coffelt's copyrighted work derives one pixel in only about 20 CPU clock cycles.

14. Autodesk is an American corporation which makes software for the architecture, engineering, construction, manufacturing, media, and entertainment industries.

15. For 35 consecutive years, between the year 1982 through about 2010, all Autodesk products use ray tracing.

16. For 35 consecutive years, between the year 1982 through about 2010, Autodesk's products use polygon surface approximations with ray tracing.

17. For 35 consecutive years, between the year 1982 through about 2010, Autodesk's products

create polygon surface shading approximations with ray tracing.

18. For 35 consecutive years, between the year 1982 through about 2010, Autodesk's products create 2 dimensional shadow approximations on polygon surface approximations with ray tracing.

19. For 35 consecutive years, between the year 1982 through about 2010, Autodesk's products create non-photorealistic digital images.

20. Sony Imageworks is making unauthorized derivative works of Coffelt's copyrighted works. Sony Imageworks makes and distributes a product titled Open Source Shading Language (OSL); also referred to as ("imageworks/OpenShadingLanguage") on public internet sites. Exhibits attached to this complaint show Sony Imageworks has derived their OSL source code from Coffelt's copyrighted works. OSL is Not a ("staple article or commodity of commerce suitable for substantial noninfringing use"). Surface shading created by OSL is identical to surface shading created by Coffelt's copyrighted works.

21. Sony Imageworks publications confirm that Autodesk's products are adapted to distribute the infringing OSL source code.

22. Autodesk publications confirm that Autodesk has adapted their products to distribute OSL. The unauthorized distribution of OSL by Sony Imageworks is direct copyright infringement. Therefore, Autodesk materially contributes to the direct copyright infringement of Coffelt's copyrighted works.

23. In April, 2017, Coffelt notified Autodesk of the alleged copyright infringement. These two components, substantial contribution, and knowledge of infringing activity, show Autodesk is liable for contributory infringement of Coffelt's copyrighted works.

24. In July, 2017, Coffelt notified all Autodesk executives of the alleged copyright infringement. All Autodesk executives have an explicit reason to know distribution of OSL is copyright infringement. Each Autodesk executive, as an individual, is inducing the copyright infringement of Coffelt's claimed works. These two components, inducing infringement, and knowledge of infringing activity, show each Autodesk executive, as an individual, is liable for contributory infringement of Coffelt's copyrighted works.

25. Autodesk has the right and ability to supervise and control the distribution of OSL.

Autodesk is the sole owner of all rights title and interest in their asserted software products. Furthermore, Autodesk's Software License Terms expressly provide that Autodesk may cancel any user's access to OSL at any time.

26. Autodesk obtains a direct benefit from the unauthorized distribution of OSL. Autodesk obtains this benefit through license fees for their software products. Autodesk also advertises the awesome photorealistic results created by their shaders, which is OSL.

27. These two components, right and ability to control, and direct benefit, show Autodesk has vicarious liability in the copyright infringement claims of this action.

28. Discovery in this action will show that Autodesk is either committing acts of direct copyright infringement, contributory infringement, or has vicarious liability, in regard to Coffelt's Vector Work.

29. Discovery in this action will show that Autodesk is either committing acts of direct copyright infringement, contributory infringement, or has vicarious liability, in regard to Coffelt's Steradian Work.

30. Coffelt's CAD Work comprises a combination of Coffelt's Vector Work, Coffelt's Gradient Work, and Coffelt's Steradian Work. Therefore, Discovery will show Autodesk is either committing acts of direct copyright infringement, contributory infringement, or has vicarious liability, in regard to Coffelt's CAD Work.

31. In April, 2017, Coffelt contacted Autodesk CEO Carl Bass, Pixar, and Nvidia, Corporation regarding the issue of photorealistic CAD. Coffelt requested each separately to explain how they are creating photorealistic digital images. Furthermore, Coffelt notified Autodesk, Pixar, and Nvidia of Coffelt's copyrighted computer programs. To this date, there has been no reply to Coffelt's request for information in regard to photorealistic CAD.

32. In June, 2017, Coffelt sent a Cease and Desist letter to Autodesk CEO, Carl Bass, in regard to the present copyright infringement issues.

33. In July, 2017, Coffelt sent a Cease and Desist letter to each executive officer and director of Autodesk in regard to the present copyright infringement issues.

34. In February, 2013, Autodesk and Sony Imageworks has access to Coffelt's copyrighted work through Coffelt's U.S. patent No. 8,614,710 publication.

35. On 3 occurrence, first in the year 2010, second in the year 2011, and third in the year 2013, Autodesk attains access to Coffelt's copyrighted works by California Department of Corrections (CDC) agents. CDC agents have caused Coffelt's copyrighted works to be copied and distributed world wide without Coffelt's authorization.

36. Forty years of failed attempts, a dissection of Coffelt's solution, documents showing distribution, and identical results, is substantial evidence showing Defendant Autodesk is committing acts in violation of 17 U.S.C. § 501, is liable for contributory copyright infringement, or has vicarious liability, directed to Coffelt's copyrighted works.

37. The copyright infringement claims herein are not exhaustive. Coffelt will file additional copyright infringement actions against Autodesk and specific individuals.

STATEMENT OF FACTS

38. Plaintiff Coffelt is the sole owner of all rights title and interest in Federally Registered Copyrights of Coffelt's creative works. The following is a list of Coffelt's registered copyrights, including and not limited to: *(all Exhibits are in the attached Appendix)*

Coffelt's Copyrighted Works

39. On December 14, 2017, Coffelt filed an application for United States copyright for Coffelt's work titled "Vector Plane Intersection" Registration No. TXu002035517 registration date: December 14, 2016 (Vector Work) *See* EXHIBIT 100. Coffelt's Vector Work source code is attached in EXHIBIT 100. Coffelt's Vector Work was created in the year 2010.

40. On May 13, 2017, Coffelt filed an application for United States copyright for Coffelt's work titled "Realistic 3D Surface Shading by Reflective Intensity 2010" case number 1-5121154211 (Gradient Work 2010) *See* EXHIBIT 101. Coffelt's Gradient Work 2010 source code is attached in EXHIBIT 101. Coffelt's Gradient Work 2010 was created in the year 2010.

41. On June 12, 2017, Coffelt filed an application for United States copyright for Coffelt's work titled "Photorealistic Surface Shading by Reflective Intensity 2017" case number 1-5376971191 (Photorealistic Gradient Work) *See* Coffelt's source code in EXHIBIT 102.

Coffelt's Photorealistic Gradient Work was created in the year 2013; and is a derivative work of Coffelt's Gradient Work 2010. Coffelt's Photorealistic Gradient Work appears in Coffelt's Gradient Work.

42. On December 13, 2016, Coffelt filed an application for United States copyright for Coffelt's work titled "CAD Reflective Intensity" Registration No. TXu002049564 registration date: December 13, 2016 (Gradient Work) *See* EXHIBIT 103. Coffelt's Gradient Work source code is attached in EXHIBIT 103. Coffelt's Gradient Work was created in the year 2013; and is a derivative work of Coffelt's Gradient Work 2010.

43. On December 15, 2016, Coffelt filed an application for United States copyright for Coffelt's work titled "Steradian Space For Light Occlusion Derivation" Registration No. TX0008356641 registration date: December 15, 2016 (Steradian Work) *See* EXHIBIT 104. Coffelt's Steradian Work source code is attached in EXHIBIT 104. Coffelt's Steradian Work was created in the year 2010.

44. On December 28, 2016, Coffelt filed an application for United States copyright for Coffelt's work titled "emoshaGraphics CAD alpha" registration No. TXu002037997 registration date: December 28, 2016 (CAD Work Alpha) *See* EXHIBIT 105. Coffelt's CAD Work Alpha comprises a combination of Coffelt's Vector Work, Coffelt's Gradient Work, and Coffelt's Steradian Work. The first 3 pages of Coffelt's CAD Work Alpha is attached in EXHIBIT 105.

45. On January 13, 2017, Coffelt filed an application for United States copyright for Coffelt's work titled "emoshaGraphics CAD" Registration No. TX0008400276 registration date: January 13, 2017 (CAD Work) *See* EXHIBIT 106. Coffelt's CAD Work comprises a combination of Coffelt's Vector Work, Coffelt's Gradient Work, and Coffelt's Steradian Work. The first 3 pages of Coffelt's CAD Work is attached in EXHIBIT 106.

Coffelt's Particular Results

46. Coffelt's Vector Work is a Literary Work; comprising a computer program creating particular results comprising:

(a) On Saturday, November 16, 2013, a specific distinct set of bytes in Coffelt's computer which correspond to a specific distinct cylinder graphic object, having graphical photorealistic

resolution in a CAD scene *See* EXHIBIT 107;

(b) On Thursday, November 14, 2013, a specific distinct set of bytes in Coffelt's computer which correspond to a specific distinct sphere graphic object, having graphical photorealistic resolution in a CAD scene *See* EXHIBIT 108;

(c) On Friday, September 20, 2013, a specific distinct set of bytes in Coffelt's computer which correspond to a specific distinct plane graphic object, having graphical photorealistic resolution in a CAD scene *See* EXHIBIT 109;

(d) On Saturday, November 16, a specific photorealistic image of the cylinder graphical object on Coffelt's computer monitor *See* EXHIBIT 107;

(e) On Thursday, November 14, 2013, a specific photorealistic image of the sphere graphical object on Coffelt's computer monitor *See* EXHIBIT 108;

(f) On On Friday, September 20, 2013, a specific photorealistic image of the plane graphical object on Coffelt's computer monitor *See* EXHIBIT 109;

(g) A photorealistic image of the cylinder on paper *See* EXHIBIT 107;

(h) A photorealistic image of the sphere on paper *See* EXHIBIT 108;

(i) A photorealistic image of the plane on paper *See* EXHIBIT 109.

47. Coffelt's Steradian Work is a Literary Work; comprising a computer program creating particular results comprising:

(a) On Saturday, November 16, 2013 a specific distinct set of bytes in Coffelt's computer which correspond to specific distinct shadows cast onto a cylinder graphic object, having photorealistic resolution, in a CAD scene *See* EXHIBIT 107;

(b) On Thursday, November 14, 2013, a specific distinct set of bytes in Coffelt's computer, which correspond to specific distinct shadows cast onto a sphere graphic object, having graphical photorealistic resolution, in a CAD scene *See* EXHIBIT 108;

(c) On Friday, September 20, 2013, a specific distinct set of bytes in Coffelt's computer which correspond to specific distinct shadows cast onto a plane graphic object, having graphical photorealistic resolution, in a CAD scene *See* EXHIBIT 109;

(d) On Saturday, November 16, 2013, specific photorealistic shadows cast onto the cylinder

graphical object on Coffelt's computer monitor *See* EXHIBIT 107;

(e) On Thursday, November 14, 2013, specific photorealistic shadows cast onto the sphere graphical object on Coffelt's computer monitor *See* EXHIBIT 108;

(f) On On Friday, September 20, 2013, specific photorealistic shadows cast onto the plane graphical object on Coffelt's computer monitor *See* EXHIBIT 109;

(g) Photorealistic shadows cast onto the cylinder on paper *See* EXHIBIT 107;

(h) Photorealistic shadows cast onto the sphere on paper *See* EXHIBIT 108;

(i) Photorealistic shadows cast onto the plane on paper *See* EXHIBIT 109.

48. Coffelt's Gradient Work is a Literary Work; comprising a computer program creating particular results comprising:

(a) On Saturday, November 16, 2013 a specific distinct set of bytes in Coffelt's computer which correspond to specific distinct gradient on a cylinder graphic object, having photorealistic resolution, in a CAD scene *See* EXHIBIT 107;

(b) On Thursday, November 14, 2013, a specific distinct set of bytes in Coffelt's computer, which correspond to specific distinct gradient on a sphere graphic object, having graphical photorealistic resolution, in a CAD scene *See* EXHIBIT 108;

(c) On Friday, September 20, 2013, a specific distinct set of bytes in Coffelt's computer which correspond to specific distinct gradient on a plane graphic object, having graphical photorealistic resolution, in a CAD scene *See* EXHIBIT 109;

(d) On Saturday, November 16, 2013, a specific photorealistic image of the cylinder graphical object on Coffelt's computer monitor *See* EXHIBIT 107;

(e) On Thursday, November 14, 2013, specific photorealistic surface gradient on the sphere graphical object on Coffelt's computer monitor *See* EXHIBIT 108;

(f) On On Friday, September 20, 2013, specific photorealistic shadows cast onto the plane graphical object on Coffelt's computer monitor *See* EXHIBIT 109;

(g) Photorealistic shadows cast onto the cylinder on paper *See* EXHIBIT 107;

(h) Photorealistic shadows cast onto the sphere on paper *See* EXHIBIT 108;

(i) Photorealistic shadows cast onto the plane on paper *See* EXHIBIT 109.

49. Coffelt's CAD Work is a Literary Work; comprising the Vector Work, Gradient Work, Photorealistic Gradient Work, and Steradian Work having particular result comprising publications on the well known website YouTube, and having a title:

(a) SteelBallsX

https://www.youtube.com/watch?v=UJWXeHVvJu0

published November 27, 2013;

(b) emoshaGraphics (TM) CAD demo Jan 24, 2017

https://www.youtube.com/watch?v=Qfm-vxMeRMI&t=2s

published January 24, 2017.

The Foundation of Coffelt's Photorealistic CAD

50. Coffelt's Vector Work is a foundation of Photorealistic CAD. *See* EXHIBIT 100. A specific location of one pixel in a bitmap is derived by only about 10 lines of source code iterated only one time. e.g. a specific pixel row, and pixel column is derived. Vector Plane Intersection is disclosed in Coffelt's U.S. patent No. 8,614,710 (710 patent) *See* EXHIBIT 110. A search of USPTO.gov and Copyright.gov database shows Coffelt is the sole person, which discloses a concise method for vector plane intersection. Coffelt's Vector Work provides that CAD surfaces can be derived at ANY desired resolution, with 100 percent accuracy.

The Foundation of CAD 1970 through 2010

51. For more than 40 years, from about 1970 through about 2010, Computer Aided Design is based on the well-known method of "ray tracing". More than 200 lines of computer code is iterated millions of times in order to derive one pixel color in a bitmap image.

52. In the early years of CAD, server farms were developed containing thousands of servers in order to create one frame of a complex graphic scene. The improvement of computer processors eliminated the need for these server farms. However, the fundamental structure of ray tracing remains unchanged to this date. The core structure of ray tracing includes the following:

(a) utilize a particular set of pixels on a view plane (image plane) e.g. a set of pixels for a 1920 x 1080 bitmap is equal to 1920 pixel width * 1080 pixel height = 2073600 pixels;

(b) utilize a particular method to select the start location of a ray;

(c) incrementing the ray into the graphic object scene;

(d) at each ray increment, test for an intersection with each and every possible point of graphic objects (e.g. millions of graphic object points are possible).

53. Ray tracing uses flat polygons to approximate a real curved surface.

54. There are at least 380 United States Patents directed to methods for ray tracing. From 1970 through present, the core structure of ray tracing, identified above at items (a) through (d) in paragraph 52, has remain unchanged. The following patents are the results of a search of USPTO.gov patent collection data base search for the terms:

ccl/345/422 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/424 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/426 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/427 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/428 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/441 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/442 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/581 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/586 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/589 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/591 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/593 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/622 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/632 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/633 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/634 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")
ccl/345/653 and (ttl/"ray tracing" or spec/"ray tracing" or ttl/"raytracing" or spec/"raytracing")

55. There are 380 U.S. patents directed to improvements in ray tracing. For example, determine specific locations to position a ray; super sampling; using specific probability

formulas; hierarchy; and secondary rays, including others.

56. The following 380 U.S. patents are directed to ray tracing, and improvements to ray tracing; and each one of these 380 U.S. patents is incorporated herein by reference (Incorporated Patents):

9,035,945	9,024,972	9,007,388	8,988,465	8,988,449	8,988,433	8,976,199	8,970,626
8,970,592	8,970,591	8,963,918	8,952,977	8,952,961	8,933,967	8,928,658	8,907,950
8,878,873	8,872,824	8,860,733	8,860,712	8,854,369	8,854,367	8,836,702	8,823,708
8,817,014	8,797,324	8,797,322	8,791,951	8,773,422	8,760,450	8,749,552	8,736,610
8,717,366	8,698,806	8,692,828	8,675,022	8,665,271	8,659,591	8,638,332	8,629,881
8,619,094	8,619,079	8,619,078	8,593,459	8,587,588	8,581,926	8,570,322	8,564,589
8,553,028	8,547,374	8,542,231	8,520,021	8,502,819	8,493,383	8,482,561	8,466,919
8,441,482	8,436,852	8,421,821	8,421,801	8,417,261	8,411,088	8,390,618	8,379,030
8,379,026	8,379,022	8,373,715	8,373,699	8,368,694	8,363,053	8,358,305	8,355,019
8,350,846	8,339,398	8,319,825	8,310,481	8,300,049	8,284,195	8,275,397	8,274,530
8,269,770	8,259,105	8,259,101	8,253,753	8,248,416	8,248,415	8,248,412	8,248,405
8,248,401	8,243,073	8,237,730	8,237,711	8,218,903	8,217,931	8,212,816	8,212,806
8,207,968	8,203,559	8,189,006	8,189,003	8,189,001	8,188,997	8,188,996	8,179,566
8,164,590	8,160,391	8,159,499	8,159,492	8,139,060	8,134,556	8,134,551	8,130,244
8,120,991	8,120,609	8,115,763	8,106,921	8,106,906	8,102,391	8,089,481	8,085,267
8,081,185	8,077,183	8,072,454	8,063,902	8,049,752	8,035,641	8,031,210	8,031,191
8,026,915	8,018,457	8,013,857	8,009,176	7,991,240	7,983,788	7,978,192	7,973,790
7,969,433	7,952,583	7,952,574	7,940,266	7,932,913	7,932,905	7,924,295	7,903,113
7,884,819	7,880,743	7,864,187	7,864,174	7,852,336	7,830,379	7,808,501	7,808,500
7,796,128	7,791,602	7,773,087	7,768,524	7,755,628	7,755,627	7,737,974	7,737,970
7,719,544	7,719,532	7,710,431	7,692,647	7,688,320	7,652,666	7,619,626	7,609,264
7,593,019	7,589,729	7,586,489	7,573,475	7,554,540	7,548,238	7,542,044	7,525,543
7,515,152	7,499,053	7,495,664	7,479,962	7,479,960	7,471,301	7,456,837	7,446,777
7,439,973	7,432,935	7,427,996	7,414,624	7,379,060	7,358,971	7,345,687	7,324,116

7,321,370 7,310,098 7,289,119 7,286,971 7,268,789 7,256,782 7,250,948 7,246,045
7,245,301 7,233,337 7,230,624 7,230,623 7,227,555 7,218,322 7,212,207 7,199,795
7,196,704 7,184,042 7,173,622 7,170,510 7,167,177 7,154,504 7,148,891 7,136,790
7,133,044 7,133,041 7,129,944 7,129,942 7,126,605 7,123,259 7,113,184 7,106,325
7,102,636 7,098,915 7,084,871 7,079,157 7,079,139 7,071,938 7,071,936 7,050,054
7,050,053 7,047,014 7,046,243 7,034,825 7,034,818 7,027,046 7,012,615 7,012,604
7,002,589 7,002,570 6,999,096 6,989,832 6,985,240 6,983,082 6,982,714 6,979,084
6,972,758 6,961,058 6,956,570 6,943,805 6,943,789 6,940,529 6,940,508 6,933,939
6,924,816 6,922,193 6,919,909 6,909,436 6,864,890 6,828,978 6,825,851 6,798,409
6,791,567 6,788,304 6,784,882 6,781,598 6,771,272 6,753,878 6,731,304 6,731,284
6,724,393 6,724,384 6,704,017 6,697,062 6,646,640 6,639,597 6,628,298 6,597,359
6,583,787 6,570,578 6,567,083 6,556,200 6,515,664 6,512,995 6,496,597 6,466,227
6,466,207 6,437,796 6,434,278 6,429,864 6,421,050 6,414,684 6,414,681 6,400,365
6,400,364 6,373,485 6,369,818 6,359,629 6,348,919 6,342,889 6,329,989 6,329,988
6,324,347 6,323,863 6,307,568 6,300,965 6,285,376 6,268,863 6,226,005 6,222,937
6,157,387 6,157,385 6,128,021 6,111,582 6,097,854 6,097,394 6,064,393 6,061,065
6,044,181 6,034,691 6,016,150 6,009,190 5,987,164 5,986,668 5,966,134 5,966,131
5,940,067 5,936,630 5,933,146 5,903,274 5,823,780 5,821,942 5,809,219 5,796,407
5,742,796 5,742,293 5,729,672 5,717,848 5,715,384 5,687,307 5,684,937 5,673,376
5,638,499 5,602,979 5,594,854 5,594,850 5,594,844 5,588,098 5,583,975 5,566,283
5,553,214 5,550,959 5,548,693 5,528,741 5,528,737 5,526,471 5,488,700 5,384,901
5,384,899 5,371,778 5,355,442 5,313,568 5,305,430 5,299,298 5,297,043 5,283,859
5,257,355 5,239,624 5,138,699 5,058,042 5,038,302 5,031,117 5,025,400 4,987,554
4,928,250 4,865,423 4,807,158 4,645,459

CAD Surface Gradients 1970 through 2010

57. For more than 40 years, starting in about the year 1970 through about 2010, CAD programs used ray tracing and a series of flat polygons to approximate curved surfaces. (polygon approximation). For example, polygon approximation uses a specific quantity of triangular surface area to define one portion of a curved surface. Polygon approximation does not create realistic surfaces. Polygon approximation is inherently described in the Incorporated Patents.

58. Vector Plane intersection equations do not exist prior to Coffelt's Vector Work. These Incorporated Patents are a basis. Prior CAD explicitly use ray tracing.

59. Polygon approximation curved surfaces are not realistic. Therefore, realistic surface gradients can not exist on these curved surface approximations. For more than 40 years, starting in about the year 1970 through about 2010, all CAD surface gradients are non-realistic approximations. *See* EXHIBIT 111. The AutoCAD drawing in EXHIBIT 111 is exemplary of all prior CAD non-realistic surface shading; source title: ("AutoCAD 2009 and AutoCAD LT 2009: No Experience Required") By Jon McFarland Internet search results are replete with monotone surfaces allegedly created by AutoCAD 2009.

60. Pixar results in EXHIBIT 112 through EXHIBIT 119 show evidence of State of the Art of CAD. EXHIBIT 112 through EXHIBIT 119 show photorealistic CAD images begin about the year 2013.

61. Polygon approximation surfaces are not realistic. Therefore, realistic shadows can not exist on these surface approximations. For more than 40 years, starting in about the year 1970 through about 2010, all CAD shadows are non-realistic approximations. *See* United States District Court, for the Central District of California, case No. ED CV16-00457 Coffelt v Nvidia, Doc. No. 38, Doc. No. 41 which is incorporated herein by reference; *See* United States Court of Appeals for the Federal Circuit case No. 17-1119 Doc. 2, Filed: 11/08/2016 which is incorporated herein by reference; *See* United States Court of Appeals for the Federal Circuit, Coffelt v. Nvidia, case No. 17-1119 Doc. 21, 22 which is incorporated herein by reference.

Coffelt's Photorealistic CAD Surface Gradients

62. Coffelt's CAD Work creates results which are significantly distinct from all prior CAD results. Coffelt's CAD Work uses Coffelt's Vector Work. Coffelt's Vector Work creates photorealistic surfaces. *See* EXHIBIT 107, EXHIBIT 108, EXHIBIT 109. Coffelt's CAD Work creates photorealistic images

63. Coffelt's Gradient Work is explained in EXHIBIT 120. In EXHIBIT 120, page 1, shows: a graphic surface (S); a reflection vector (rfla); a reflection vector (rflb); a view point VP; a light source point SP; a maximum distance to the View Point (d0a); a minimum distance to the View Point d0b; and a surface normal N. EXHIBIT 120 shows a core component of Coffelt's Gradient Work which is the surface gradient is based on the angle between the reflection vector and the direction of view. There are 2 directions of view in EXHIBIT 120, view vector (vpa), and view vector (vpb). There are 2 angles in EXHIBIT 120 used to derive the photorealistic gradient. Angle (a) is the angle between vector (rfla) and vector (vpa); (a) is the maximum angle for all reflection vectors on surface (S). Angle (b) is the angle between vector (rflb) and vector (vpb); (b) is the minimum angle for all reflection vectors on surface (S).

64. EXHIBIT 120, page 2, shows a linear equation used to derive the photorealistic surface gradient. The quantity of color shift is derived by the linear equation shown in EXHIBIT 120 page 2. In this example, (-50) is the maximum color shift, and zero is the minimum color shift. The maximum color shift occurs at (maxd); and the minimum shift occurs at (mind). The point slope equation is derived from the given values of : (-50), (mind), and (maxd). During runtime of Coffelt's Gradient Work, many various angles will be derived between the view direction and the reflection vector. The quantity of color shift is derived by this linear equation for each distinct d0 (or cosine of the angle), or other equivalent parameter.

65. The foregoing paragraphs 63 and 64 is a technical description of the foundation of Coffelt's Gradient Work, and Photorealistic Gradient Work; and the corresponding source code is shown in the following paragraph 66.

Coffelt's Foundation of Photorealistic Gradients

66. The foundation of Coffelt's Gradient Work, and Photorealistic Gradient Work is shown in Coffelt's source code in lines 0000 through 0014 as follows (TXu002049564 EXHIBIT 102):

```
0000  rflx = rptx - ptx00a;                                                    (A)
0001  rfly = rpty - pty00a;                                                    (A)
0002  rflz = rptz - ptz00a;                                                    (A)
0003  lenrfl = sqrt(rflx * rflx + rfly * rfly + rflz * rflz);                  (B)
0004  vpdotrfl = (vpax * rflx + vpay * rfly + vpaz * rflz) / (lenvpa * lenrfl); (B)
0005  theta = acos(vpdotrfl);                                                  (C)
0006  mgrad = -50 / (max_d - min_d);          (D) (E) (F)
0007  d0 = lenvpa * sin(theta);               (G)
0008  shiftd = mgrad * (d0 - min_d);          (H)
0009  blueD = 100.0;                          (I)
0010  greenD = 255.0;                         (I)
0011  redD = 100.0;                           (I)
0012  blueD += shiftd;                        (J)
0013  greenD += shiftd;                       (J)
0014  redD += shiftd;                         (J)
```

67. Larry Gritz dissects the foregoing Coffelt's source code, lines 0000 through 0014, and distributes it into various files in order to hide copyright infringement. Therefore, the above item reference (A) through (J) are used to identify the corresponding location of infringing OSL source code in EXHIBIT 121.

68. OSL source code is available for download to the public at the following url: https://github.com/imageworks/OpenShadingLanguage *See* EXHIBIT 122.

69. On Tuesday, August 01, 2017, 6:16:42 PM, Coffelt downloaded a copy of OSL source code from: https://github.com/imageworks/OpenShadingLanguage (OSL Infringing Code). The OSL Infringing Code is shown in EXHIBIT 121.

Open Source Shading Language Copyright Infringement

70. Sony Pictures Imageworks, having an office address at 9050 W. Washington Blvd. Culver City, CA 90232 (Sony Imageworks) makes the product OSL source code.

71. The OSL source code in EXHIBIT 121 is identical to Coffelt's Gradient Work, or is a derivative of Coffelt's Gradient Work *See* EXHIBIT 121. For these reasons shown in EXHIBIT 121, OSL is an unauthorized derivative of Coffelt's Gradient Work. For these reasons, Sony Imageworks have infringed Coffelt's copyright in the identified OSL Products, in violation of Section 501 of the Copyright Act, 17 U.S.C. § 501(a).

72. The OSL source code in EXHIBIT 121 is identical to Coffelt's Gradient Work, or is a derivative of Coffelt's Gradient Work *See* EXHIBIT 121. Sony Imageworks distributes this infringing OSL source code at website: https://github.com/imageworks/OpenShadingLanguage *See* EXHIBIT 122. For these reasons, Sony Imageworks is, without authorization, distributing this infringing OSL source code. For these reasons, Sony Imageworks have infringed Coffelt's copyright in the identified OSL Products, in violation of Section 501 of the Copyright Act, 17 U.S.C. § 501(a).

Larry Gritz, Open Source Shading Language

73. Open Source Shading Language (OSL) by Sony Imageworks and Larry Gritz (Gritz) is allegedly a new programming language. *See* EXHIBIT 127. A review of OSL source code shows OSL is merely a C++ language Application Program Interface (API) *See* EXHIBIT 121. A unique variable name "closure" is purportedly the basis for OSL being a new language. Gritz confirms that OSL is Not a new language by his allegation: ("it is simply a more convenient notation for describing shading") *See* EXHIBIT 127.
source: http://blenderdiplom.com/en/interviews/531-interview-larry-gritz-lead-developer-of-osl.html
date EXHIBIT 127 downloaded: November 23, 2017.

74. Gritz does not provided any technical explanation of how OSL is a new language. Gritz only discloses that a "closure" is new, without any technical explanation of the physical structure of a "closure". *See* EXHIBIT 127.

75. Gritz explains that the programing language C is "clunky", and is the cause for the

necessity for a new programing language. Gritz does not explain the meaning of "clunky"; and does not explain how he has overcome the problem of "clunky". Gritz merely explains the awesome results of his allegedly new programming language. *See* EXHIBIT 127.

76. Gritz also explains that it is optimal for Sony Imageworks to give away valuable software free, rather than keep it to themselves. *See* EXHIBIT 127.

77. Larry Gritz is awarded a technical achievement award for OSL, without any any explanation of his technical achievement. Larry Gritz receives an Academy Award for only the photorealistic results. *See* EXHIBIT 128, and EXHIBIT 129.

78. Blender publications confirm that OSL is based on Coffelt's Gradient Work. Source: https://docs.blender.org/manual/es/dev/render/cycles/nodes/types/input/fresnel.html date: August 13, 2017 *See* EXHIBIT 123. In EXHIBIT 123, the Blender publication alleges that the photorealistic surface gradient is based on the direction of view, and the reflection vector.

79. A BusinessWire publication confirms that OSL is used in Autodesk Beast. Source: http://www.businesswire.com/news/home/20130325005005/en/Autodesk-Reveals-New-Gameware-Advancements-GDC-2013 date: November 23, 2017 *See* EXHIBIT 130.

80. A Sony Imageworks publication confirms that OSL is used in Autodesk Beast. Source: http://opensource.imageworks.com/ date: August 16, 2017 *See* EXHIBIT 131.

Autodesk, Inc.

81. Autodesk has adapted their products AutoCad, Fusion 360, Maya, InfraWorks, AutoCAD Civil 3D, Revit, Inventor, or Beast (Autodesk Products), to materially contribute to acts of copyright infringement of Coffelt's copyrighted works.

82. Autodesk publication confirms that Autodesk has made unauthorized copies of Coffelt's copyrighted works. *See* EXHIBIT 134; Source:

http://help.autodesk.com/view/BEAST/2015/ENU/?guid=__files_GUID_B812FA2F_A188_4D9A_A5A8_ACD7A771AA89_htm

date: November 23, 2017 (Autodesk OSL Publication).

83. The Autodesk OSL Publication shows Autodesk has adapted their product Maya or Beast to download OSL source code identified in EXHIBIT 121:

("These attributes are used to set up the BeastOSL node in Maya.") *See* EXHIBIT 134 at page 4. Furthermore, the Autodesk OSL Document directs a person to the url:

https://github.com/imageworks/OpenShadingLanguage (Infringing URL)

See EXHIBIT 134 at page 1.

84. This Infringing URL published in the Autodesk OSL Document confirms that Autodesk has adapted Beast or Maya to download the OSL source code shown in EXHIBIT 121. EXHIBIT 121 at page 1 shows the Infringing URL.

85. The Autodesk OSL Publication also shows Autodesk's Application Program Interface (Autodesk's OSL API) for OSL. *See* EXHIBIT 134 at pages 2, 3.

86. Autodesk publication confirms that Autodesk has adapted their product Beast to download the asserted OSL source code; publication title: ("What's New in Beast 2013.2.x"); Source:

https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-08210E19-206D-4643-96EE-24DFDEE68845-htm.html

date: November 23, 2017 *See* EXHIBIT 132.

87. Autodesk publication confirms that Autodesk has adapted their product Beast to download the asserted OSL source code; publication title: ("New Feature: Open Shading Language Support"); Source:

https://forums.autodesk.com/t5/beast/new-feature-open-shading-language-support/td-p/4285745

date: August 11, 2017 *See* EXHIBIT 133.

88. Autodesk product Maya or Beast software is purchased by a person (User). During operation of Maya or Beast software by the User, the User's computer is required to download a copy of OSL source code in order to execute a shading function: ("These attributes are used to set up the BeastOSL node in Maya.") *See* EXHIBIT 134 page 4. Autodesk has specifically adapted Maya or Beast to execute this User's download of OSL. Therefore, this User's computer is required to execute a distribution of OSL source code.

89. A standard software development procedure is to compile and test the source code of the project. This test is typically used to make corrections to the software, and is commonly referred to as fixing bugs. The test are used to ensure the software produce the intended results.

90. For the reasons in paragraphs 82 through 89 inclusive, Autodesk has executed a test of the Autodesk OSL API. For the reasons in paragraphs 82 through 89 inclusive, Autodesk executed a test of Autodesk's Maya, and executed a test of Beast.

91. OSL is Not a staple article or commodity of commerce suitable for substantial noninfringing use. Autodesk publications and Sony Imageworks publications in this complaint are replete with evidence that the sole purpose of OSL is a shader function in CAD software. OSL is not stand alone software. e.g. OSL does not create the physical structure of a graphic object. OSL only creates the color of the graphic object. OSL must be used with other CAD software.

92. For the reasons in paragraphs 70 through 72; 82 through 91, Autodesk has downloaded and created an unauthorized copy of OSL in order to complete testing of the Autodesk OSL API, Maya, or Beast.

93. For the reasons in paragraphs 70 through 72; 82 through 92 Autodesk has made unauthorized copies of Coffelt's copyrighted Gradient Work, TXu002049564, or Photorealistic Gradient Work, case No. 1-5376971191.

94. For the reasons in paragraphs 70 through 72; 82 through 93, Autodesk materially contributes to the unauthorized distribution of Coffelt's copyrighted Gradient Work, registration No. TXu002049564, or Photorealistic Gradient Work, case No. 1-5376971191.

95. In April, 2017, Coffelt notified Autodesk of the alleged copyright infringement.

96. There are many reasons Autodesk had reason to know OSL is directed to copyright infringement, including, not limited to:

(a) Coffelt's notice to Autodesk of the alleged copyright infringement in April, 2017;

(b) 40 years of failed attempts by all others to create photorealistic digital images;

(c) Autodesk's 35 years of failed attempts to create photorealistic digital images;

(d) The numerous failed attempts to create photorealistic digital images,

e.g. including the 380 Incorporated Patents;

(e) Autodesk is one of about only 3 worldwide major CAD software developers;

(f) Coffelt is the sole individual claiming intellectual property directed to photorealistic CAD.

e.g Coffelt is the sole author of Vector Plane Intersection; and complex 3D shadows

97. One who knowingly induces, causes, or materially contributes to copyright infringement, by another but who has not committed or participated in the infringing acts himself may be held liable as a contributory infringer if he or she had knowledge, or reason to know, of the infringement. See Metro-Goldwyn-Mayer Studios Inc. v Grokster, Ltd. 545 U.S. 913 (2005); Sony Corp. v Universal City Studios, Inc. 464 U.S. 417 (1984).

98. For the reasons in paragraphs 70 through 72; 82 through 97, Autodesk is committing acts of contributory infringement of Coffelt's copyrighted Gradient Work, registration No. TXu002049564, or Photorealistic Gradient Work, case No. 1-5376971191.

99. Autodesk is the sole owner of all rights title and interest in their CAD software products, including Beast and Maya. Autodesk makes the asserted software products Beast and Maya. Autodesk has the right and ability to remove source code from Beast and Maya, which provides access to OSL. Autodesk has both the right an ability to stop Maya and Beast from causing any infringing activity. Therefore, Autodesk has the right and ability to supervise the infringing activity identified in paragraphs 70 through 73 inclusive, and paragraphs 82 through 96 inclusive.

100. Autodesk offers a license for their product Maya on the public internet for $1470.00 per year. *See* EXHIBIT 135.

Source: https://www.autodesk.com/products/maya/overview date: November 26, 2017.

101. Autodesk has received financial benefit from the license of Maya or Beast.

102. When the right and ability to supervise coalesce with an obvious and direct financial interest in the exploitation of copyrighted materials. The purposes of copyright law may be best effectuated by the imposition of liability upon the beneficiary of that exploitation. See Shapiro, Bernstein & Co. v H.L. Green Co. 316 F.2d 304, 307 (2d Cir. 1963).

103. For the reasons in paragraphs 70 through 72; 82 through 91; 99 through 102, Autodesk has vicarious liability in this action directed to Coffelt's copyrighted Gradient Work, registration No. TXu002049564, or Photorealistic Gradient Work, case No. 1-5376971191.

OSL Results

104. OSL creates photorealistic surfaces in CAD. OSL creates results identical to Coffelt's CAD Work results. A comparison in EXHIBIT 136 shows Autodesk's surface shading is identical to Coffelt's surface shading on the cylinder. Autodesk uses OSL to create the surface shading on the cylinder. Coffelt uses Coffelt's Vector Work, Gradient Work, and CAD Work to create the surface shading on the cylinder. EXHIBIT 137 through EXHIBIT 141 show additional examples that OSL results are identical to Coffelt's results.

Autodesk Access To Coffelt's Copyrighted Work

105. Autodesk attained access to Coffelt's copyrighted works on February 28, 2013 by Coffelt's U.S patent No. 8,614,710 publication.

106. On about March 18, 2010, at 428 Devener Street, Apt. # C, Riverside, CA 92507 California Department of Corrections (CDC) agents forcefully took copies of Coffelt's Work. Autodesk has a significant relationship with CDC. Evidence of this significant relationship will be provided to this Court.

107. In about the year 2011, at 1195 Spring Street, Apt. # C Riverside, CA 92507 CDC agents forcefully took copies of Coffelt's Work. Evidence of this unauthorized copy of Coffelt's copyrighted work will be provided to this Court.

108. In about the year 2013, at 14327 Frederick Street, Moreno Valley, CA 92553 CDC agents forcefully took copies of Coffelt's Work. Evidence of this unauthorized copy of Coffelt's copyrighted work will be provided to this Court.

109. For the above reasons, Autodesk attained access to Coffelt's copyrighted works on at least 3 occurrence, first in the year 2010, second about the year 2011, and third, about the year 2013.

110. Coffelt served the following individuals with a Cease and Desist letter directed to the copyright infringement issues in this action:

Autodesk Executives:

Andrew Anagnost, Carl Bass, Crawford W. Beveridge, Steve Blum, Chris Bradshaw,

Moonhie Chin, Pascal W. DiFronzo, Reid French, Thomas Georgens, R. Scott Herren, Richard S. Hill, Jeff Kowalski, Mary T. McDowell, Lorrie M. Norrington, Elizabeth Rafael, Stacy J. Smith, Eric Mitchel, Will Harris, Jorge Garcia, Edwin Robledo;

GitHub individuals:

Alison Marcozzi, Chris Wanstrath (Corporate Executive Officer).

111. Larry Gritz is not authorized to copy or distribute Coffelt's copyrighted works. Sony Imageworks is not authorized to copy or distribute Coffelt's copyrighted works. Autodesk is not authorized to copy or distribute Coffelt's copyrighted works.

112. Coffelt has Not authorized any rights, in Coffelt's copyrighted works.

113. Coffelt has Not authorized any title, in Coffelt's copyrighted works.

114. Coffelt has Not authorized any interest, in Coffelt's copyrighted works.

FIRST CAUSE OF ACTION

(Copyright Infringement – 17 U.S.C. §501)

115. Plaintiff repeats and incorporates by this reference the allegations set forth in paragraphs 1 through 114, inclusive.

116. Plaintiff Coffelt is the author and sole owner of all rights title and interest of the claimed works copied by Autodesk through various products including without limitation, AutoCad, Fusion 360, Maya, InfraWorks, AutoCAD Civil 3D, Revit, Inventor, or Beast.

117. For each of the claimed works in this matter, Plaintiff holds a copyright registration certificate from the United States Copyright Office.

118. Without authorization, Autodesk copied the following Plaintiff owned and copyrighted claimed work including:

(i) "Photorealistic Surface Shading by Reflective Intensity 2017", Case No. 1-5376971191, (ii) "Realistic 3D Surface Shading by Reflective Intensity 2010", Case No. 1-5121154211, or (iii) "CAD Reflective Intensity" registration No. TXu002049564.

119. Through their conduct averred herein, Defendants have infringed Plaintiffs' copyright in the above identified Autodesk Products, in violation of Section 501 of the Copyright Act,

17 U.S.C. § 501(a).

120. Defendants' acts of infringement are willful, intentional and purposeful, in disregard of and with indifference to Plaintiff's rights.

121. As a direct and proximate result of said infringement by Defendants, Plaintiff is entitled to damages of at least $33,000,000 to be proven at trial.

122. Plaintiff is also entitled to Defendant's profits attributable to the infringement, pursuant to 17 U.S.C. § 504(b), including an accounting of such profits.

123. Plaintiff is further are entitled to Plaintiff's attorneys' fees and full costs pursuant to 17 U.S.C. § 505 and otherwise according to law.

124. As a direct and proximate result of the foregoing acts and conduct, Plaintiff has sustained and will continue to sustain substantial, immediate, and irreparable injury, for which there is no adequate remedy at law. Plaintiff is informed and believe and on that basis aver that unless enjoined and restrained by this Court, Defendants will continue to infringe Plaintiff's rights in the Infringed Works. Plaintiff is entitled to preliminary and permanent injunctive relief to restrain and enjoin Defendants' continuing infringing conduct.

SECOND CAUSE OF ACTION

(Copyright Infringement – Contributory Infringement)

125. Plaintiff repeats and incorporates by this reference the allegations set forth in paragraphs 1 through 124, inclusive.

126. Plaintiff Coffelt is the author and sole owner of all rights title and interest of the claimed works distributed by Autodesk's products including without limitation, AutoCad, Fusion 360, Maya, InfraWorks, AutoCAD Civil 3D, Revit, Inventor, or Beast.

127. For each of the claimed works in this matter, Plaintiff holds a copyright registration certificate from the United States Copyright Office.

128. Without authorization, Autodesk adapted, including without limitation, AutoCad, Fusion 360, Maya, InfraWorks, AutoCAD Civil 3D, Revit, Inventor, or Beast, to distribute the following Plaintiff owned and copyrighted claimed work including:

(i) "Photorealistic Surface Shading by Reflective Intensity 2017", Case No. 1-5376971191,

(ii) "Realistic 3D Surface Shading by Reflective Intensity 2010", Case No. 1-5121154211, or (iii) "CAD Reflective Intensity" registration No. TXu002049564.

129. Through their conduct averred herein, Defendants have infringed Plaintiffs' copyright by contributory infringement.

130. Defendants' acts of infringement are willful, intentional and purposeful, in disregard of and with indifference to Plaintiff's rights.

131. As a direct and proximate result of said infringement by Defendants, Plaintiff is entitled to damages of at least $33,000,000,000 to be proven at trial.

132. Plaintiff is also entitled to Defendant's profits attributable to the infringement, pursuant to 17 U.S.C. § 504(b), including an accounting of such profits.

133. Plaintiff is further are entitled to Plaintiff's attorneys' fees and full costs pursuant to 17 U.S.C. § 505 and otherwise according to law.

134. As a direct and proximate result of the foregoing acts and conduct, Plaintiff has sustained and will continue to sustain substantial, immediate, and irreparable injury, for which there is no adequate remedy at law. Plaintiff is informed and believe and on that basis aver that unless enjoined and restrained by this Court, Defendants will continue to infringe Plaintiff's rights in the Infringed Works. Plaintiff is entitled to preliminary and permanent injunctive relief to restrain and enjoin Defendants' continuing infringing conduct.

THIRD CAUSE OF ACTION

(Copyright Infringement – Vicarious Liability)

135. Plaintiff repeats and incorporates by this reference the allegations set forth in paragraphs 1 through 134, inclusive.

136. Plaintiff Coffelt is the author and sole owner of all rights title and interest of the claimed works distributed by Autodesk's products including without limitation, AutoCad, Fusion 360, Maya, InfraWorks, AutoCAD Civil 3D, Revit, Inventor, or Beast.

137. For each of the claimed works in this matter, Plaintiff holds a copyright registration certificate from the United States Copyright Office.

138. Without authorization, Autodesk adapted, including without limitation, AutoCad, Fusion

360, Maya, InfraWorks, AutoCAD Civil 3D, Revit, Inventor, or Beast, to distribute the following Plaintiff owned and copyrighted claimed work including:

(i) "Photorealistic Surface Shading by Reflective Intensity 2017", Case No. 1-5376971191,

(ii) "Realistic 3D Surface Shading by Reflective Intensity 2010", Case No. 1-5121154211, or

(iii) "CAD Reflective Intensity" registration No. TXu002049564.

139. Through their conduct averred herein, Defendants have infringed Plaintiffs' copyright by vicarious liability.

140. Defendants' acts of infringement are willful, intentional and purposeful, in disregard of and with indifference to Plaintiff's rights.

141. As a direct and proximate result of said infringement by Defendants, Plaintiff is entitled to damages of at least $22,000,000,000 to be proven at trial.

142. Plaintiff is also entitled to Defendant's profits attributable to the infringement, pursuant to 17 U.S.C. § 504(b), including an accounting of such profits.

143. Plaintiff is further are entitled to Plaintiff's attorneys' fees and full costs pursuant to 17 U.S.C. § 505 and otherwise according to law.

144. As a direct and proximate result of the foregoing acts and conduct, Plaintiff has sustained and will continue to sustain substantial, immediate, and irreparable injury, for which there is no adequate remedy at law. Plaintiff is informed and believe and on that basis aver that unless enjoined and restrained by this Court, Defendants will continue to infringe Plaintiff's rights in the Infringed Works. Plaintiff is entitled to preliminary and permanent injunctive relief to restrain and enjoin Defendants' continuing infringing conduct.

FOURTH CAUSE OF ACTION

(Copyright Infringement – Vicarious Liability)

145. Plaintiff repeats and incorporates by this reference the allegations set forth in paragraphs 1 through 144, inclusive.

146. Plaintiff Coffelt is the author and sole owner of all rights title and interest of the claimed works distributed by Autodesk's products including without limitation, AutoCad, Fusion 360, Maya, InfraWorks, AutoCAD Civil 3D, Revit, Inventor, or Beast.

147. For each of the claimed works in this matter, Plaintiff holds a copyright registration certificate from the United States Copyright Office.

148. Without authorization, Autodesk adapted, including without limitation, AutoCad, Fusion 360, Maya, InfraWorks, AutoCAD Civil 3D, Revit, Inventor, or Beast, to distribute the following Plaintiff owned and copyrighted claimed work including:

(i) "Vector Plane Intersection" registration No. TXu002035517, or

(ii) "Steradian Space For Light Occlusion Derivation" registration No. TX0008356641.

149. Through their conduct averred herein, Defendants have infringed Plaintiffs' copyright by vicarious liability.

150. Defendants' acts of infringement are willful, intentional and purposeful, in disregard of and with indifference to Plaintiff's rights.

151. As a direct and proximate result of said infringement by Defendants, Plaintiff is entitled to damages of at least $11,000,000,000 to be proven at trial.

152. Plaintiff is also entitled to Defendant's profits attributable to the infringement, pursuant to 17 U.S.C. § 504(b), including an accounting of such profits.

153. Plaintiff is further are entitled to Plaintiff's attorneys' fees and full costs pursuant to 17 U.S.C. § 505 and otherwise according to law.

154. As a direct and proximate result of the foregoing acts and conduct, Plaintiff has sustained and will continue to sustain substantial, immediate, and irreparable injury, for which there is no adequate remedy at law. Plaintiff is informed and believe and on that basis aver that unless enjoined and restrained by this Court, Defendants will continue to infringe Plaintiff's rights in the Infringed Works. Plaintiff is entitled to preliminary and permanent injunctive relief to restrain and enjoin Defendants' continuing infringing conduct.

FIFTH CAUSE OF ACTION

(Copyright Infringement – 17 U.S.C. §501)

155. Plaintiff repeats and incorporates by this reference the allegations set forth in paragraphs 1 through 154, inclusive.

156. Plaintiff Coffelt is the author and sole owner of all rights title and interest of the claimed

works distributed by Sony Imageworks through various products including without limitation, Open Source Shading Language (OSL).

157. For each of the claimed works in this matter, Plaintiff holds a copyright registration certificate from the United States Copyright Office.

158. Without authorization, Sony Imageworks distributed the following Plaintiff owned and copyrighted claimed work including:

(i) "Photorealistic Surface Shading by Reflective Intensity 2017", Case No. 1-5376971191, (ii) "Realistic 3D Surface Shading by Reflective Intensity 2010", Case No. 1-5121154211, or (iii) "CAD Reflective Intensity" registration No. TXu002049564.

159. Through their conduct averred herein, Sony Imageworks have infringed Plaintiffs' copyright in Open Source Shading Language (OSL), in violation of Section 501 of the Copyright Act, 17 U.S.C. § 501(a).

160. Sony Imageworks acts of infringement are willful, intentional and purposeful, in disregard of and with indifference to Plaintiff's rights.

RELIEF

161. WHEREFORE, Plaintiff request the following judgement against Defendant Autodesk as follows:

162. For Plaintiff's damages in the amount of $ 33,000,000,000 (thirty three billion dollars) and any additional damages proven at trial; and Defendant's profits;

163. For preliminary and permanent injunction enjoining Defendant Autodesk and all persons acting in concert or participation with Autodesk from (a) directly or indirectly reproducing, distributing, or otherwise infringing in any manner on Plaintiff's copyrighted works.

164. For Plaintiff's attorneys' fees and full costs incurred in this action.

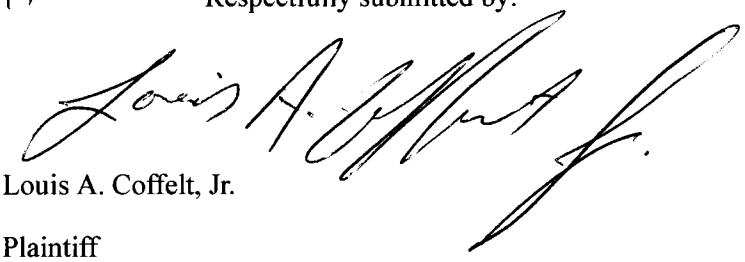
165. For any additional relief as this Court may deem just and proper.

DEMAND FOR JURY TRIAL

Plaintiff, Louis A. Coffelt, Jr., hereby request a jury trial for all issues raised in this action.

Date: November 30, 2017

Respectfully submitted by:



Louis A. Coffelt, Jr.

Plaintiff

Pro Se

APPENDIX

EXHIBIT 100

Certificate of Registration

This Certificate issued under the seal of the Copyright Office in accordance with title 17, *United States Code*, attests that registration has been made for the work identified below. The information on this certificate has been made a part of the Copyright Office records.

Acting United States Register of Copyrights and Director

Registration Number
TXu 2-035-517
Effective Date of Registration:
December 14, 2016

Title

Title of Work: Vector Plane Intersection

Title of Larger Work: emoshaGraphics CAD

Completion/Publication

Year of Completion: 2013

Author

- **Author:** Louis Arthur Coffelt
  Author Created: computer program
  Citizen of: United States
  Domiciled in: United States
  Year Born: 1959

Copyright Claimant

Copyright Claimant: Louis Arthur Coffelt
231 E. Alessandro Blvd., 6A-504, Riverside, CA, 92508, United States

Rights and Permissions

Name: Louis Arthur Coffelt
Email: louis.coffelt@gmail.com
Telephone: (951)790-6086
Address: 231 E. Alessandro Blvd., 6A-504
Riverside, CA 92508 United States

Certification

Name: Louis Arthur Coffelt, Jr.
Date: December 14, 2016
Applicant's Tracking Number: cad1122

```
void ImageAndPanelCls::IntersectijPlane(double &scrnxP, double &scrnyP, double ptxP, double ptyP, double ptzP,
double rptxP, double rptyP, double rptzP)
{
   si = ptxP - rptxP;
   sj = ptyP - rptyP;
   sk = ptzP - rptzP;
   i = abs(si);
   j = abs(sj);
   k = abs(sk);
   s0i = ptxP;
   s0j = ptyP;
   s0k = ptzP;
   if (i > 0.000000001)
   {
      mji = sj / si;
      mki = sk / si;
      tempi = (mki * s0i - s0k) / mki;
      tempj = mji * (tempi - s0i) + s0j;
   }
   else if (j > 0.000000001)
   {
      mij = si / sj;
      mkj = sk / sj;
      tempj = (mkj * s0j - s0k) / mkj;
      tempi = mij * (tempj - s0j) + s0i;
   }
   else if (k > 0.000000001)
   {
      mik = si / sk;
      mjk = sj / sk;
      tempi = mik * (-s0k) + s0i;
      tempj = mjk * (-s0k) + s0j;
   }
      scrnxP = tempi;
      scrnyP = tempj;
}
//
void ImageAndPanelCls::IntersectjkPlanePartialSolution(double &jP, double &kP, double rptxP, double rptyP, double
rptzP, double ptxP, double ptyP, double ptzP)
{
   si = rptxP - ptxP;
   sj = rptyP - ptyP;
   sk = rptzP - ptzP;
      //
   k = abs(sk);
      //
      s0i = ptxP;
      s0j = ptyP;
   s0k = ptzP;
   if (k > 0.000000001)
   {
      mik = si  / sk;
      mjk = sj / sk;
```


COFFELT'S VECTOR WORK SOURCE CODE TXu002035517

```
      tempk = (mik * s0k - s0i) / mik;
      tempj = mjk * (tempk - s0k) + s0j;
   }
      jP = tempj;
      kP = tempk;
}
//
void ImageAndPanelCls::IntersectAnyPlanePartialSolution(double &ixintersectionP, double &jxintersectionP, double
&kxintersectionP, double N1iP, double N1jP, double N1kP, double N0iP, double N0jP, double N0kP, double ptxP,
double ptyP, double ptzP, double rptxP, double rptyP, double rptzP)
{
      N1ic = N1iP;
      N1jc = N1jP;
      N1kc = N1kP;
      N0ic = N0iP;
      N0jc = N0jP;
      N0kc = N0kP;
      Nic = N1ic - N0ic;
      Njc = N1jc - N0jc;
      Nkc = N1kc - N0kc;
   si = ptxP - rptxP;
   sj = ptyP - rptyP;
   sk = ptzP - rptzP;
   i = abs(si);
   j = abs(sj);
   k = abs(sk);
   s0i = ptxP;
   s0j = ptyP;
   s0k = ptzP;
   if (k > 0.000000001)
   {
      mik = si / sk;
      mjk = sj / sk;
      tempk = (mik * s0k * Nic - s0i * Nic + N0ic * Nic + mjk * s0k * Njc - s0j * Njc + N0jc * Njc + N0kc * Nkc) /
(mik * Nic + mjk * Njc + Nkc);
      tempi = mik * (tempk - s0k) + s0i;
            tempj = mjk * (tempk - s0k) + s0j;
   }
      ixintersectionP = tempi;
      jxintersectionP = tempj;
      kxintersectionP = tempk;
}
//
```

EXHIBIT 101

```
C:\Documents and Settings\louis\My ...\VecPlnInt\ColorByReflectionVec.cs 1
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace VecPlnInt
{
        class ColorByReflectionVec
        {
                double ColorDouble = 0.0;
                int ColorInt = 0;
                //
                public void SetColorByReflectionVec(ref int RedComponentP, ref int GreenComponentP, ref int
        BlueComponentP,ref int TotalChanges, double AdotBp1,
         int BaseColorRedP, int BaseColorGreenP, int BaseColorBlueP)
                {
                        // turquoise is approx 2f f2 f1
                        // decreasing blue shifts towards green
                        // decreasing green shifts towards royal blue (dark)
                        // use 2 points of Reflection vector, and 2 points (view pt, surface pt)
                        vector, cos of angle between these two vectors to set color
                        // if adotb < 0 : darken color
                        // total shift is 3 red w/ 16 blue
                        // 3 blue w/ 16 red
                        // 3 green w/ 16 blue
                        // 3 blue w/ 16 green
                        //ShiftLimit1 = 255 - ShiftNum01;
                        //ShiftLimit2 = 255 - ShiftNum02;
                        //ShiftLimit3 = 255 - ShiftNum03;
                        if (AdotBp1 > 0)
                        {
                                ColorDouble = 224 * AdotBp1;
                                ColorInt = (int)ColorDouble;
                                GreenComponentP = ColorInt;
                        }
                        Else
                        {
                                GreenComponentP = 0;
                        }
                        RedComponentP = BaseColorGreenP;
                        BlueComponentP = BaseColorBlueP;
                        TotalChanges++;
                }
        }
}
```

Reproduction of original file

Coffelt's Gradient Work 2010
Original Computer File Location: Coffelt's Laptop, Service Tag GMBTY32
File name: exhibit_102_realistic_3d_surface_shading_2010.pdf
Modified / Original Created: Wednesday, October 20, 2010, 8:01:16 AM

U.S. Copyright Application No. 1-5121154211

EXHIBIT 102

```
//
//
//
//
0000    rflx = rptx - ptx00a;
0001    rfly = rpty - pty00a;
0002    rflz = rptz - ptz00a;
0003    lenrfl = sqrt(rflx * rflx + rfly * rfly + rflz * rflz);
0004    vpdotrfl = (vpax * rflx + vpay * rfly + vpaz * rflz) / (lenvpa * lenrfl);
0005    theta = acos(vpdotrfl);
0006    mgrad =  -50 / (max_d - min_d);
0007    d0 = lenvpa * sin(theta);
0008    shiftd = mgrad * (d0 - min_d);
0009    blueD = 100.0;
0010    greenD = 255.0;
0011    redD = 100.0;
0012    blueD += shiftd;
0013    greenD += shiftd;
0014    redD += shiftd;
//
//
//
//
// Copyright 2017 by Louis A. Coffelt, Jr.
// TITLE OF THIS WORK: Photorealistic Surface Shading by Reflective Intensity 2017
// TYPE OF WORK: Computer Program
//
// This work is based on the Work by Louis A. Coffelt, Jr. created on
// Wednesday, October 20, 2010, 8:01:16 AM titled:
// ("Realistic 3D Surface Shading by Reflective Intensity 2010 ")
// Application Date: 5/13/2017.
// Service Request #: 1-5121154211
//
// This work is used in a larger work titled ("CAD Reflective Intensity")
// Application Date: December 13, 2016
// Service Request #: 1-4249380951
//
// The Claimed Work is the c++ program above at lines 0000 through 0014
// A description of this Claimed Work is the following:
//
//
//
//
// An objective of this claimed Work includes to derive photorealistic 3D surface
// shading for any type surface. A 3D graphic object is identified by a
// mathematical equation. There is a View Point in the 3D scene. There is a point
// light source in the 3D scene. Light source Incident Vectors intersect the
// graphic object. Light is reflected from the graphic object (Reflected Vector).
// The angle of incidence is equal to the reflected angle. The Reflected Vector and
// View Point are used to derive the light intensity for each corresponding point on
// the graphic object.
//
//
// lines 0000 through 0002 are the x, y, z, components of the Reflected Vector rfl.
// line 0003, lenrfl is the length of the Reflected Vector rfl.
// line 0004, vpa is the vector between the View Point and the graphic object point.
// line 0004, vpdotrfl is the vector dot product of vpa and rfl (cosine of angle).
// line 0005, theta is the angle between vectors vpa and rfl.
// line 0006, max_d is the maximum distance between the View Point and vector rfl.
// line 0006, min_d is the minimum distance between the View Point and vector rfl.
// line 0006, -50 is a selected constant for maximum shift of the base surface color.
// line 0006, mgrad is slope of a linear equation, which derives the color shift value.
// line 0007, d0 is current distance between the View Point and vector rfl.
// line 0008, shiftd is the value of the color shift from the base color value.
// lines 0009 through 0011, the base color of the graphic object surface is assigned.
// lines 0012 through 0014, the base color is shifted in order to create the
//                      photorealistic surface shading gradient.
```

COFFELT's Photorealistic Gradient Work Source Code

U.S. Application No. 1-5376971191

EXHIBIT 103

Certificate of Registratio..

This Certificate issued under the seal of the Copyright Office in accordance with title 17, *United States Code*, attests that registration has been made for the work identified below. The information on this certificate has been made a part of the Copyright Office records.

Acting United States Register of Copyrights and Director

Registration Number
TXu 2-049-564
Effective Date of Registration:
December 13, 2016

Title

Title of Work: CAD Reflective Intensity

Title of Larger Work: emoshaGraphics CAD

Completion/Publication

Year of Completion: 2013

Author

- **Author:** Louis Arthur Coffelt
  Author Created: computer program
  Citizen of: United States
  Domiciled in: United States
  Year Born: 1959

Copyright Claimant

Copyright Claimant: Louis Arthur Coffelt
231 E. Alessandro Blvd., 6A-504, Riverside, CA, 92508, United States

Rights and Permissions

Name: Louis Arthur Coffelt
Email: louis.coffelt@gmail.com
Telephone: (951)790-6086
Address: 231 E. Alessandro Blvd., 6A-504
Riverside, CA 92508 United States

Certification

Name: Louis Arthur Coffelt, Jr.
Date: December 13, 2016
Applicant's Tracking Number: cad1133

```
#include "StdAfx.h"
#include "Objects_Cls.h"
#include <cmath>

void Objects_Cls::Sphere_____________Steradians__(cli::array<double, 1>^ &StrDistV01p, cli::array<double, 1>^
&VisibleV01p, int systemNum01P, int numSurface01P, double t0p)
{
    __int64 SizeSTRv = StrDistV01p->Length;
    __int64 SizeVISv = VisibleV01p->Length;
    short int systemNum = systemNum01P;
    short int numSurface = numSurface01P;
    t0 = t0p;
    SetDynamicData(systemNum);
    double lampSphereR = 0.4;
    double mainSphereR = 0.5;
    double ptx00a = 0.0;
    double pty00a = 0.0;
    double ptz00a = 0.0;
    double ptx02a = 0.0;
    double pty02a = 0.0;
    double ptz02a = 0.0;
    int scrnindx = 0;
    int scrncolx = 0;
    int scrnrowx = 0;
    double scrncolxD = 0.0;
    double scrnrowxD = 0.0;
    double scrnxA = 0.0;
    double scrnyA = 0.0;
    double scrnxC = 0.0;
    double scrnyC = 0.0;
    double cVisDistA = 0.0;
    double pVisDistA = 0.0;
    int StrIndxPxA = 0;
    double cStrDistA = 0.0;
    double pStrDistA = 0.0;
    //
    double d0 = 0.0;
    double N0000x = 0.0;
    double N0000y = 0.0;
    double N0000z = 0.0;
    double N0100x = 0.0;
    double N0100y = 0.0;
    double N0100z = 0.0;
    double N00x = 0.0;
    double N00y = 0.0;
    double N00z = 0.0;
    double N0002x = 0.0;
    double N0002y = 0.0;
    double N0002z = 0.0;
    double N0102x = 0.0;
    double N0102y = 1.0;
    double N0102z = 0.0;
    NextCoordinatesType3conversionOnly(N0000x, N0000y, N0000z, N0002x, N0002y, N0002z);
```


COFFELT'S GRADIENT WORK SOURCE CODE TXu002049564

```
NextCoordinatesType3conversionOnly(N0100x, N0100y, N0100z, N0102x, N0102y, N0102z);
N00x = N0100x - N0000x;
N00y = N0100y - N0000y;
N00z = N0100z - N0000z;
double spax = 0.0;
double spay = 0.0;
double spaz = 0.0;
double spdx = 0.0;
double spdy = 0.0;
double spdz = 0.0;
double vpax = 0.0;
double vpay = 0.0;
double vpaz = 0.0;
double bx = 0.0;
double by = 0.0;
double bz = 0.0;
double cx = 0.0;
double cy = 0.0;
double cz = 0.0;
double spacos = 1.0;
double spasin = 1.0;
double dx = 0.0;
double dy = 0.0;
double dz = 0.0;
double ex = 0.0;
double ey = 0.0;
double ez = 0.0;
double rptx = 0.0;
double rpty = 0.0;
double rptz = 0.0;
double rflx = 0.0;
double rfly = 0.0;
double rflz = 0.0;
double spadotc = 1.0;
double vpdotrfl = 1.0;
double phi = 0.0;
double phix = 0.0;
double thetax = 0.0;
double theta = 0.0;
double c0 = 1.0;
double lenc = 1.0;
double lend = 1.0;
double lene = 1.0;
double lenspa = 1.0;
double lenvpa = 1.0;
double lenrfl = 1.0;
//
double pi = 3.1415926;
double twopi = 2.0 * pi;
double pid2 = pi / 2.0;
double threepid2 = 3.0 * pi / 2.0;
double pid6 = pi / 6.0;
double fivepid6 = 5.0 * pi / 6.0;
double sevenpid6 = 7.0 * pi / 6.0;
```

```
double elevenpid6 = 11.0 * pi / 6.0;
double r0 = 1.5;
double lenxy = 0.07;
double xz_limit = 0.0;
double m_s = 0.0;
double x = 0.0;
double y = 0.0;
double z = 0.0;
if (systemNum == 850 || systemNum == 851 || systemNum == 852 || systemNum == 853)
{
    dphi_c = dphi_850;
}
while (phix < twopi)
{
    thetax = theta_i;
    while (thetax < twopi)
    {
        pty02a = r_sphere_ui * cos(thetax);
        ptz02a = r_sphere_ui * sin(thetax) * sin(phix);
        ptx02a = r_sphere_ui * sin(thetax) * cos(phix);
        N0102x = ptx02a;
        N0102y = pty02a;
        N0102z = ptz02a;
        NextCoordinatesType3conversionOnly(N0100x, N0100y, N0100z, N0102x, N0102y, N0102z);
        N00x = N0100x - N0000x;
        N00y = N0100y - N0000y;
        N00z = N0100z - N0000z;
        NextCoordinatesType3(ptx00a, pty00a, ptz00a, scrnxA, scrnyA, ptx02a, pty02a, ptz02a);
        if (scrnxA > 0.0 && scrnxA < scrnWinches && scrnyA > 0.0 && scrnyA < scrnHinches)
        {
            scrncolxD = scrnxA * scrnppiD;
            scrnrowxD = scrnyA * scrnppiD;
            scrncolx = int(scrncolxD);
            scrnrowx = int(scrnrowxD);
            scrnindx = scrnrowx * scrnWpx + scrncolx;
            if (scrnindx < SizeVISv)
            {
                pVisDistA = VisibleV01p[scrnindx];
                cVisDistA = sqrt(si * si + sj * sj + sk * sk);
                if (cVisDistA < pVisDistA)
                {
                    VisibleV01p[scrnindx] = cVisDistA;
                    //
                    spax = spx - ptx00a;
                    spay = spy - pty00a;
                    spaz = spz - ptz00a;
                    lenspa = sqrt(spax * spax + spay * spay + spaz * spaz);
                    vpax = vpx - ptx00a;
                    vpay = vpy - pty00a;
                    vpaz = vpz - ptz00a;
                    lenvpa = sqrt(vpax * vpax + vpay * vpay + vpaz * vpaz);
                    bx = N00y * spaz - spay * N00z;
                    by = -(N00x * spaz - spax * N00z);
                    bz = N00x * spay - spax * N00y;
```

```
cx = N00y * bz - by * N00z;
cy = -(N00x * bz - bx * N00z);
cz = N00x * by - bx * N00y;
lenc = sqrt(cx * cx + cy * cy + cz * cz);
spadotc = (spax * cx + spay * cy + spaz * cz) / (lenspa * lenc);
phi = acos(spadotc);
spacos = lenspa * abs(cos(phi));
spasin = lenspa * sin(phi);
c0 = spacos / lenc;
ex = c0 * cx;
ey = c0 * cy;
ez = c0 * cz;
dx = ptx00a - ex;
dy = pty00a - ey;
dz = ptz00a - ez;
spdx = spx - dx;
spdy = spy - dy;
spdz = spz - dz;
rptx = ptx00a + ex + spdx;
rpty = pty00a + ey + spdy;
rptz = ptz00a + ez + spdz;
rflx = rptx - ptx00a;
rfly = rpty - pty00a;
rflz = rptz - ptz00a;
lenrfl = sqrt(rflx * rflx + rfly * rfly + rflz * rflz);
vpdotrfl = (vpax * rflx + vpay * rfly + vpaz * rflz) / (lenvpa * lenrfl);
theta = acos(vpdotrfl);
d0 = lenvpa * sin(theta);
if (systemNum == 850 && d0 < min_d_850)
{
    min_d_850 = d0;
}
if (systemNum == 850 && d0 > max_d_850)
{
    max_d_850 = d0;
}
if (systemNum == 851 && d0 < min_d_851)
{
    min_d_851 = d0;
}
if (systemNum == 851 && d0 > max_d_851)
{
    max_d_851 = d0;
}
if (systemNum == 852 && d0 < min_d_852)
{
    min_d_852 = d0;
}
if (systemNum == 852 && d0 > max_d_852)
{
    max_d_852 = d0;
}
if (systemNum == 853 && d0 < min_d_853)
{
```

```
                                min_d_853 = d0;
                            }
                            if (systemNum == 853 && d0 > max_d_853)
                            {
                                max_d_853 = d0;
                            }
                        }
                    }
                }
                thetax += dphi_c;
            }
            phix += dphi_c;
        }
        int stophere = 0;
}
//
void Objects_Cls::Sphere_____________Iteration__(cli::array<System::Byte, 1>^ &RedVp, cli::array<System::Byte,
1>^ &GreenVp, cli::array<System::Byte, 1>^ &BlueVp, cli::array<double, 1>^ StrDistV00p, cli::array<double, 1>^
VisibleV00p, int systemNum00P, int numSurface00P, int numColor00P)
{
    __int64 sizeRedV = RedVp->Length;
    __int64 SizeSTRv = StrDistV00p->Length;
    __int64 SizeVISv = VisibleV00p->Length;
    int numColor = numColor00P;
    short int systemNum = systemNum00P;
    short int numSurface = numSurface00P;
    SetDynamicData(systemNum);
    double blueD = 180.0;
    double greenD = 180.0;
    double redD = 180.0;
    int btBlueInt = 180;
    int btGreenInt = 180;
    int btRedInt = 180;
    double shiftd = 0.0;
    double mgrad = -222.0 / 10.0;
    double ptx00b = 0.0;
    double pty00b = 0.0;
    double ptz00b = 0.0;
    double ptx02b = 0.0;
    double pty02b = 0.0;
    double ptz02b = 0.0;
    double spx00 = spx;
    double spy00 = spy;
    double spz00 = spz;
    double spx02 = 0.0;
    double spy02 = 0.0;
    double spz02 = 0.0;
    double vpx00 = vpx;
    double vpy00 = vpy;
    double vpz00 = vpz;
    double vpx02 = 0.0;
    double vpy02 = 0.0;
    double vpz02 = 0.0;
    int scrnindx = 0;
```

```
int scrncolx = 0;
int scrnrowx = 0;
double scrncolxD = 0.0;
double scrnrowxD = 0.0;
double scrnxB = 0.0;
double scrnyB = 0.0;
double cVisDistB = 0.0;
double pVisDistB = 0.0;
int StrIndxPxB = 0;
double cStrDistB = 0.0;
double pStrDistB = 0.0;
double scrnxD = 0.0;
double scrnyD = 0.0;
double cVisDistD = 0.0;
double pVisDistD = 0.0;
int StrIndxPxD = 0;
double cStrDistD = 0.0;
double pStrDistD = 0.0;
double deltaStr = 1.0;
double deltaVis = 1.0;
double pid3 = 3.1415926 / 3.0;
double d0 = 0.0;
double N0000x = 0.0;
double N0000y = 0.0;
double N0000z = 0.0;
double N0100x = 0.0;
double N0100y = 0.0;
double N0100z = 0.0;
double N00x = 0.0;
double N00y = 0.0;
double N00z = 0.0;
double N0002x = 0.0;
double N0002y = 0.0;
double N0002z = 0.0;
double N0102x = 0.0;
double N0102y = 1.0;
double N0102z = 0.0;
NextCoordinatesType3conversionOnly(N0000x, N0000y, N0000z, N0002x, N0002y, N0002z);
NextCoordinatesType3conversionOnly(N0100x, N0100y, N0100z, N0102x, N0102y, N0102z);
N00x = N0100x - N0000x;
N00y = N0100y - N0000y;
N00z = N0100z - N0000z;
double spax = 0.0;
double spay = 0.0;
double spaz = 0.0;
double spdx = 0.0;
double spdy = 0.0;
double spdz = 0.0;
double vpax = 0.0;
double vpay = 0.0;
double vpaz = 0.0;
double bx = 0.0;
double by = 0.0;
double bz = 0.0;
```

```
double cx = 0.0;
double cy = 0.0;
double cz = 0.0;
double spacos = 1.0;
double spasin = 1.0;
double dx = 0.0;
double dy = 0.0;
double dz = 0.0;
double ex = 0.0;
double ey = 0.0;
double ez = 0.0;
double rptx = 0.0;
double rpty = 0.0;
double rptz = 0.0;
double rflx = 0.0;
double rfly = 0.0;
double rflz = 0.0;
double spadotc = 1.0;
double vpdotrfl = 1.0;
double phi = 0.0;
double c0 = 1.0;
double lenc = 1.0;
double lend = 1.0;
double lene = 1.0;
double lenspa = 1.0;
double lenvpa = 1.0;
double lenrfl = 1.0;
int minRed = 1000;
int maxRed = 0;
double phix = 0.0;
double thetax = 0.0;
double theta = 0.0;
double pi = 3.1415926;
double twopi = 2.0 * pi;
double pid2 = pi / 2.0;
double threepid2 = 3.0 * pi / 2.0;
double pid6 = pi / 6.0;
double fivepid6 = 5.0 * pi / 6.0;
double sevenpid6 = 7.0 * pi / 6.0;
double elevenpid6 = 11.0 * pi / 6.0;
double r0 = 1.5;
double lenxy = 0.07;
double xz_limit = 0.0;
double m_s = 0.0;
double x = 0.0;
double y = 0.0;
double z = 0.0;
int countx = 0;
int county = 0;
System::String^ data = " ";
int maxr = 0;
int minr = 1000;
int maxg = 0;
int ming = 1000;
```

```
int maxb = 0;
int minb = 1000;
if (systemNum == 850)
{
	mgrad = -50 / (max_d_850 - min_d_850);
	dphi_c = dphi_850;
}
else if (systemNum == 851)
{
	mgrad = -50 / (max_d_851 - min_d_851);
	dphi_c = dphi_850;
}
if (systemNum == 852)
{
	mgrad = -50 / (max_d_852 - min_d_852);
	dphi_c = dphi_850;
}
else if (systemNum == 853)
{
	mgrad = -50 / (max_d_853 - min_d_853);
	dphi_c = dphi_850;
}
while (phix < twopi)
{

	thetax = theta_i;
	while (thetax < twopi)
	{
		pty02b = r_sphere_ui * cos(thetax);
		ptz02b = r_sphere_ui * sin(thetax) * sin(phix);
		ptx02b = r_sphere_ui * sin(thetax) * cos(phix);
		N0102x = ptx02b;
		N0102y = pty02b;
		N0102z = ptz02b;
		NextCoordinatesType3conversionOnly(N0100x, N0100y, N0100z, N0102x, N0102y, N0102z);
		N00x = N0100x - N0000x;
		N00y = N0100y - N0000y;
		N00z = N0100z - N0000z;
		NextCoordinatesType3(ptx00b, pty00b, ptz00b, scrnxB, scrnyB, ptx02b, pty02b, ptz02b);
		if (scrnxB > 0.0 && scrnxB < scrnWinches && scrnyB > 0.0 && scrnyB < scrnHinches)
		{
			scrncolxD = scrnxB * scrnppiD;
			scrnrowxD = scrnyB * scrnppiD;
			scrncolx = int(scrncolxD);
			scrnrowx = int(scrnrowxD);
			scrnindx = scrnrowx * scrnWpx + scrncolx;
			if (scrnindx < SizeVISv)
			{
				pVisDistB = VisibleV00p[scrnindx];
				cVisDistB = sqrt(si * si + sj * sj + sk * sk);
				deltaVis = abs(cVisDistB - pVisDistB);
				if (deltaVis < 0.001)
				{
					spax = spx - ptx00b;
```

```
spay = spy - pty00b;
spaz = spz - ptz00b;
lenspa = sqrt(spax * spax + spay * spay + spaz * spaz);
vpax = vpx - ptx00b;
vpay = vpy - pty00b;
vpaz = vpz - ptz00b;
lenvpa = sqrt(vpax * vpax + vpay * vpay + vpaz * vpaz);
bx = N00y * spaz - spay * N00z;
by = -(N00x * spaz - spax * N00z);
bz = N00x * spay - spax * N00y;
cx = N00y * bz - by * N00z;
cy = -(N00x * bz - bx * N00z);
cz = N00x * by - bx * N00y;
lenc = sqrt(cx * cx + cy * cy + cz * cz);
spadotc = (spax * cx + spay * cy + spaz * cz) / (lenspa * lenc);
phi = acos(spadotc);
spacos = lenspa * abs(cos(phi));
spasin = lenspa * sin(phi);
c0 = spacos / lenc;
ex = c0 * cx;
ey = c0 * cy;
ez = c0 * cz;
dx = ptx00b - ex;
dy = pty00b - ey;
dz = ptz00b - ez;
spdx = spx - dx;
spdy = spy - dy;
spdz = spz - dz;
rptx = ptx00b + ex + spdx;
rpty = pty00b + ey + spdy;
rptz = ptz00b + ez + spdz;
rflx = rptx - ptx00b;
rfly = rpty - pty00b;
rflz = rptz - ptz00b;
lenrfl = sqrt(rflx * rflx + rfly * rfly + rflz * rflz);
vpdotrfl = (vpax * rflx + vpay * rfly + vpaz * rflz) / (lenvpa * lenrfl);
theta = acos(vpdotrfl);
d0 = lenvpa * sin(theta);
if (systemNum == 850)
{
    shiftd = mgrad * (d0 - min_d_850);
}
if (systemNum == 851)
{
    shiftd = mgrad * (d0 - min_d_851);
}
if (systemNum == 852)
{
    shiftd = mgrad * (d0 - min_d_852);
}
if (systemNum == 853)
{
    shiftd = mgrad * (d0 - min_d_853);
}
```

```
                                blueD = 55.0;
                                greenD = 255.0;
                                redD = 55.0;
                                blueD += shiftd;
                                greenD += shiftd;
                                redD += shiftd;
                                btBlueInt = int(blueD);
                                btGreenInt = int(greenD);
                                btRedInt = int(redD);
                                countx++;
                                if (btBlueInt > 255)
                                {
                                    btBlueInt = 255;
                                }
                                if (btBlueInt < 0)
                                {
                                    btBlueInt = 0;
                                }
                                if (btGreenInt > 255)
                                {
                                    btGreenInt = 255;
                                }
                                if (btGreenInt < 0)
                                {
                                    btGreenInt = 0;
                                }
                                if (btRedInt > 255)
                                {
                                    btRedInt = 255;
                                }
                                if (btRedInt < 0)
                                {
                                    btRedInt = 0;
                                }
                                btBlue = System::Byte(btBlueInt);
                                btGreen = System::Byte(btGreenInt);
                                btRed = System::Byte(btRedInt);
                                countx++;
                                RedVp[scrnindx] = btRed;
                                GreenVp[scrnindx] = btGreen;
                                BlueVp[scrnindx] = btBlue;
                            }
                        }
                    }
                    thetax += dphi_c;
                }
                phix += dphi_c;
            }
            //
            int stophere = 0;
        }
        //
        void Objects_Cls::NextCoordinatesType3(double &ptx33P, double &pty33P, double &ptz33P, double &scrnx33P,
        double &scrny33P, double ptx33p, double pty33p, double ptz33p)
```

```
{
    ptm03 = ptx33p;
    ptn03 = pty33p;
    pto03 = ptz33p;
    ptd03 = ptm03;
    pte03 = ptn03 * ne03 + pto03 * oe03;
    ptf03 = ptn03 * nf03 + pto03 * of03;
    pti03 = ptd03 * di03 + ptf03 * fi03;
    ptj03 = pte03;
    ptk03 = ptd03 * dk03 + ptf03 * fk03;
    ptx03 = pti03 * ix03 + ptj03 * jx03;
    pty03 = pti03 * iy03 + ptj03 * jy03;
    ptz03 = ptk03;
    ptm02 = ptx03 + T03x;
    ptn02 = pty03 + T03y;
    pto02 = ptz03 + T03z;
    ptd02 = ptm02;
    pte02 = ptn02 * ne02 + pto02 * oe02;
    ptf02 = ptn02 * nf02 + pto02 * of02;
    pti02 = ptd02 * di02 + ptf02 * fi02;
    ptj02 = pte02;
    ptk02 = ptd02 * dk02 + ptf02 * fk02;
    ptx02 = pti02 * ix02 + ptj02 * jx02;
    pty02 = pti02 * iy02 + ptj02 * jy02;
    ptz02 = ptk02;
    ptm01 = ptx02 + T02x;
    ptn01 = pty02 + T02y;
    pto01 = ptz02 + T02z;
    ptd01 = ptm01;
    pte01 = ptn01 * ne01 + pto01 * oe01;
    ptf01 = ptn01 * nf01 + pto01 * of01;
    pti01 = ptd01 * di01 + ptf01 * fi01;
    ptj01 = pte01;
    ptk01 = ptd01 * dk01 + ptf01 * fk01;
    ptx01 = pti01 * ix01 + ptj01 * jx01;
    pty01 = pti01 * iy01 + ptj01 * jy01;
    ptz01 = ptk01;
    ptm00 = ptx01 + T01x;
    ptn00 = pty01 + T01y;
    pto00 = ptz01 + T01z;
    ptd00 = ptm00;
    pte00 = ptn00 * ne00 + pto00 * oe00;
    ptf00 = ptn00 * nf00 + pto00 * of00;
    pti00 = ptd00 * di00 + ptf00 * fi00;
    ptj00 = pte00;
    ptk00 = ptd00 * dk00 + ptf00 * fk00;
    ptx00 = pti00 * ix00 + ptj00 * jx00;
    pty00 = pti00 * iy00 + ptj00 * jy00;
    ptz00 = ptk00;
    ptx00 += Tcgx;
    pty00 += Tcgy;
    ptz00 += Tcgz;
    si = ptx00 - vpx;
    sj = pty00 - vpy;
```

```
	sk = ptz00 - vpz;
	i = abs(si);
	j = abs(sj);
	k = abs(sk);
	s0i = ptx00;
	s0j = pty00;
	s0k = ptz00;
	if (i > 0.000000001)
	{
		mji = sj / si;
		mki = sk / si;
		tempi = (mki * s0i - s0k) / mki;
		tempj = mji * (tempi - s0i) + s0j;
	}
	else if (j > 0.000000001)
	{
		mij = si / sj;
		mkj = sk / sj;
		tempj = (mkj * s0j - s0k) / mkj;
		tempi = mij * (tempj - s0j) + s0i;
	}
	else if (k > 0.000000001)
	{
		mik = si / sk;
		mjk = sj / sk;
		tempi = mik * (-s0k) + s0i;
		tempj = mjk * (-s0k) + s0j;
	}
	ptx33P = ptx00;
	pty33P = pty00;
	ptz33P = ptz00;
	scrnx33P = tempi;
	scrny33P = tempj;
}
//
void Objects_Cls::IntersectScreen(double &scrnxP, double &scrnyP, double ptxP, double ptyP, double ptzP)
{
	si = ptxP - vpx;
	sj = ptyP - vpy;
	sk = ptzP - vpz;
	i = abs(si);
	j = abs(sj);
	k = abs(sk);
	s0i = ptxP;
	s0j = ptyP;
	s0k = ptzP;
	if (i > 0.000000001)
	{
		mji = sj / si;
		mki = sk / si;
		tempi = (mki * s0i - s0k) / mki;
		tempj = mji * (tempi - s0i) + s0j;
	}
	else if (j > 0.000000001)
```

```
        {
            mij = si / sj;
            mkj = sk / sj;
            tempj = (mkj * s0j - s0k) / mkj;
            tempi = mij * (tempj - s0j) + s0i;
        }
        else if (k > 0.000000001)
        {
            mik = si / sk;
            mjk = sj / sk;
            tempi = mik * (-s0k) + s0i;
            tempj = mjk * (-s0k) + s0j;
        }
        scrnxP = tempi;
        scrnyP = tempj;
}
//
```

EXHIBIT 104

Certificate of Registration

This Certificate issued under the seal of the Copyright Office in accordance with title 17, *United States Code*, attests that registration has been made for the work identified below. The information on this certificate has been made a part of the Copyright Office records.

Acting United States Register of Copyrights and Director

Registration Number
TX 8-356-641
Effective Date of Registration:
December 15, 2016

Title

Title of Work: Steradian Space For Light Occlusion Derivation

Title of Larger Work: emoshaGraphics CAD

Completion/Publication

Year of Completion: 2011
Date of 1st Publication: March 14, 2016
Nation of 1st Publication: United States

Author

- **Author:** Louis Arthur Coffelt
Author Created: computer program
Citizen of: United States
Domiciled in: United States
Year Born: 1959

Copyright Claimant

Copyright Claimant: Louis Arthur Coffelt
231 E. Alessandro Blvd., 6A-504, Riverside, CA, 92508, United States

Rights and Permissions

Name: Louis Arthur Coffelt
Email: louis.coffelt@gmail.com
Telephone: (951)790-6086
Address: 231 E. Alessandro Blvd., 6A-504
Riverside, CA 92508 United States

Certification

Name: Louis Arthur Coffelt, Jr.
Date: December 15, 2016

```
// SteradiansXame.h
#include <cmath>
#pragma once
using namespace System;
namespace SteradiansXame {
	public ref class SteradianCls
	{
	public:
		static void SetSteradians(double &cmin00p, double &cmax00p, double &rmin00p, double &rmax00p, double
&strRadius00p, double &strPpiD00p, int &strColWpx00p, int &totalStrPx00p, double spx00p, double spy00p, double
spz00p, double Tcgx00p, double Tcgy00p, double Tcgz00p)
		{
			double StrPpiD = 125.0;
			double lenArcCol = 20.0;
			double lenArcRow = 20.0;
			double StrColWpxD = lenArcCol * StrPpiD;
			double StrRowHpxD = lenArcRow * StrPpiD;
			int StrColWpx = int(StrColWpxD);
			int StrRowHpx = int(StrRowHpxD);
			int totalStrPx = StrRowHpx * StrColWpx;
			double StrRadiusOffset = 4.0;
			double cgspx = Tcgx00p - spx00p;
			double cgspy = Tcgy00p - spy00p;
			double cgspz = Tcgz00p - spz00p;
			double lencgsp = sqrt(cgspx * cgspx + cgspy * cgspy + cgspz * cgspz);
			double strRadius = sqrt(cgspx * cgspx + cgspy * cgspy + cgspz * cgspz) + StrRadiusOffset;
			double totalColAngle = lenArcCol / strRadius;
			double totalRowAngle = lenArcRow / strRadius;
			double lenCgSpxy = sqrt(cgspx * cgspx + cgspy * cgspy);
			double absCgSpy = abs(cgspy);
			double cgspxydotx = cgspx / lenCgSpxy;
			double centerStrCol = acos(cgspxydotx) * cgspy / absCgSpy;
			double columnAngle = 0.5 * totalColAngle;
			double cmax = centerStrCol + columnAngle;
			double cmin = centerStrCol - columnAngle;
			double cgspdotz = cgspz / lencgsp;
			double centerStrRow = acos(cgspdotz);
			double rowAngle = 0.5 * totalRowAngle;
			double rmax = centerStrRow + rowAngle;
			double rmin = centerStrRow - rowAngle;
			cmin00p = cmin;
			cmax00p = cmax;
			rmin00p = rmin;
			rmax00p = rmax;
			strRadius00p = strRadius;
			strPpiD00p = StrPpiD;
			strColWpx00p = StrColWpx;
			totalStrPx00p = totalStrPx;
		}
	};
}
```


COFFELT'S STERADIAN WORK SOURCE CODE TX0008356641

```
void ImageAndPanelCls::NextSteradian(int &strIndxP, double &cStrDistP, double pt00xP, double pt00yP, double
pt00zP)
{
   ptx00c = pt00xP;
      pty00c = pt00yP;
      ptz00c = pt00zP;
   ptspx = ptx00c - spx;
   ptspy = pty00c - spy;
   ptspz = ptz00c - spz;
   cStrDist = sqrt(ptspx * ptspx + ptspy * ptspy + ptspz * ptspz);
   lenptspxy = sqrt(ptspx * ptspx + ptspy * ptspy);
   absPtSpy = abs(ptspy);
      if(absPtSpy < 0.0000000001)
      {
            absPtSpy = 0.0000000001;
      }
   ptspxydotx = ptspx / lenptspxy;
   StrColAngle = acos(ptspxydotx) * ptspy / absPtSpy;
   ptspdotz = ptspz / cStrDist;
   StrRowAngle = acos(ptspdotz);
   if (StrColAngle > cmin && StrColAngle < cmax && StrRowAngle > rmin && StrRowAngle < rmax)
   {
      StrColAngle -= cmin;
      StrRowAngle -= rmin;
   }
   lenArcCol = abs(strRadius * StrColAngle);
   lenArcRow = abs(strRadius * StrRowAngle);
   StrColIndxD = lenArcCol * StRppiD;
   StrRowIndxD = lenArcRow * StRppiD;
   StrColIndx = int(StrColIndxD);
   StrRowIndx = int(StrRowIndxD);
   StrIndxPx = StrRowIndx * StrColWpx + StrColIndx;
      strIndxP = StrIndxPx;
      cStrDistP = cStrDist;
}
//
```

EXHIBIT 105

Certificate of Registration

This Certificate issued under the seal of the Copyright Office in accordance with title 17, *United States Code*, attests that registration has been made for the work identified below. The information on this certificate has been made a part of the Copyright Office records.

Acting United States Register of Copyrights and Director

Registration Number
TXu 2-037-997
Effective Date of Registration:
December 28, 2016

Title

Title of Work: emoshaGraphics CAD alpha

Title of Larger Work: emoshaGraphics CAD

Completion/Publication

Year of Completion: 2016

Author

- **Author:** Louis Arthur Coffelt
Author Created: computer program
Citizen of: United States
Domiciled in: United States
Year Born: 1959

Copyright Claimant

Copyright Claimant: Louis Arthur Coffelt
231 E. Alessandro Blvd., 6A-504, Riverside, CA, 92508, United States

Rights and Permissions

Name: Louis Arthur Coffelt
Email: louis.coffelt@gmail.com
Telephone: (951)790-6086

Certification

Name: Louis Arthur Coffelt, Jr.
Date: December 28, 2016
Applicant's Tracking Number: egcad1133

```
#pragma once
#include "C:\\test\\cad_dll\\Convert_Binary_To_Doubles.h"
#include "C:\\test\\cad_dll\\WriteBinaryDoublesOrIntDLL.h"
#include "C:\\test\\cad_dll\\Convert_Doubles_To_Binary.h"
#include "Drawing_Cls.h"

namespace CppWinForm1
{

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::IO;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace Convert_Binary_To_Doubles;
	using namespace WriteBinaryDoublesDLLx;
	using namespace Convert_Doubles_To_Binary;
	/// <summary>
	/// Summary for MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{

	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::PictureBox^  fileButton;
	private: System::Windows::Forms::PictureBox^  newOpenCloseSaveButton;

	protected:

	protected:

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>
		System::ComponentModel::Container ^components;
	private: System::Windows::Forms::PictureBox^  viewButton;
	private: System::Windows::Forms::PictureBox^  rotateZoomButton;

	private: System::Windows::Forms::PictureBox^  planeButton;
```

```
private: System::Windows::Forms::PictureBox^ triangleButton;
private: System::Windows::Forms::PictureBox^ discButton;
private: System::Windows::Forms::PictureBox^ ringButton;

private: System::Windows::Forms::PictureBox^ cylinderButton;

private: System::Windows::Forms::PictureBox^ sphereButton;
private: System::Windows::Forms::PictureBox^ hemisphereButton;



private: System::Windows::Forms::PictureBox^ helpButton;
private: System::Windows::Forms::PictureBox^ outputImage;

private: System::Windows::Forms::PictureBox^ selectedSurfaceColor;
private: System::Windows::Forms::PictureBox^ deleteSurfaceButton;
private: System::Windows::Forms::PictureBox^ editSurfaceButton;
private: System::Windows::Forms::PictureBox^ colorPalletImage;
private: System::Windows::Forms::TextBox^ xp0box;
private: System::Windows::Forms::TextBox^ yp0box;
private: System::Windows::Forms::TextBox^ zp0box;
private: System::Windows::Forms::TextBox^ zp1box;
private: System::Windows::Forms::TextBox^ yp1box;
private: System::Windows::Forms::TextBox^ xp1box;
private: System::Windows::Forms::TextBox^ radiusBox;

private: System::Windows::Forms::PictureBox^ newSurfaceButton;
private: System::Windows::Forms::PictureBox^ saveSurfaceButton;
private: System::Windows::Forms::TextBox^ surfaceDescriptionBox;
private: System::Windows::Forms::PictureBox^ saveDescriptionButton;



private: System::Windows::Forms::PictureBox^ button188x32;

private: System::Windows::Forms::Label^ SelectedColorTipLabel;
private: System::Windows::Forms::Label^ recentFileName0button;

private: System::Windows::Forms::TextBox^ zp2box;
private: System::Windows::Forms::TextBox^ yp2box;
private: System::Windows::Forms::TextBox^ xp2box;
private: System::Windows::Forms::TextBox^ zp3box;
private: System::Windows::Forms::TextBox^ yp3box;
private: System::Windows::Forms::TextBox^ xp3box;
private: System::Windows::Forms::PictureBox^ lineButton;
private: System::Windows::Forms::PictureBox^ filletLinearButton;
private: System::Windows::Forms::PictureBox^ surfaceDensityButton;
private: System::Windows::Forms::PictureBox^ lightSourceButton;
private: System::Windows::Forms::TextBox^ doubleInput2box;
private: System::Windows::Forms::TextBox^ doubleInput1box;
private: System::Windows::Forms::TextBox^ doubleInput0box;
private: System::Windows::Forms::PictureBox^ saveDoublesButton;
private: System::Windows::Forms::Label^ double0Label;
private: System::Windows::Forms::Label^ double1Label;
private: System::Windows::Forms::Label^ double2Label;
private: System::Windows::Forms::PictureBox^ cancelInputButton;

private: System::Windows::Forms::PictureBox^ densityChoiceButton;
private: System::Windows::Forms::Label^ densityValueLabel;
private: System::Windows::Forms::PictureBox^ startPageButton;
private: System::Windows::Forms::PictureBox^ start_page_dwg_ico;
```

```
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  surface_type;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  Column6;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  Column7;
    private: System::Windows::Forms::PictureBox^  cancelSurfaceButton;
    private: System::Windows::Forms::PictureBox^  newProjectButton;
    private: System::Windows::Forms::Label^  recentFileName1button;
    private: System::Windows::Forms::Label^  recentFileName2button;
    private: System::Windows::Forms::Label^  recentFileName3button;









        private: System::Windows::Forms::DataGridView^  surfaceListBox;



    #pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                System::ComponentModel::ComponentResourceManager^  resources = (gcnew
    System::ComponentModel::ComponentResourceManager(MyForm::typeid));
                System::Windows::Forms::DataGridViewCellStyle^  dataGridViewCellStyle7 = (gcnew
    System::Windows::Forms::DataGridViewCellStyle());
                System::Windows::Forms::DataGridViewCellStyle^  dataGridViewCellStyle8 = (gcnew
    System::Windows::Forms::DataGridViewCellStyle());
                System::Windows::Forms::DataGridViewCellStyle^  dataGridViewCellStyle9 = (gcnew
    System::Windows::Forms::DataGridViewCellStyle());
                this->fileButton = (gcnew System::Windows::Forms::PictureBox());
                this->newOpenCloseSaveButton = (gcnew System::Windows::Forms::PictureBox());
                this->viewButton = (gcnew System::Windows::Forms::PictureBox());
                this->rotateZoomButton = (gcnew System::Windows::Forms::PictureBox());
                this->planeButton = (gcnew System::Windows::Forms::PictureBox());
                this->triangleButton = (gcnew System::Windows::Forms::PictureBox());
                this->discButton = (gcnew System::Windows::Forms::PictureBox());
                this->ringButton = (gcnew System::Windows::Forms::PictureBox());
                this->cylinderButton = (gcnew System::Windows::Forms::PictureBox());
                this->sphereButton = (gcnew System::Windows::Forms::PictureBox());
                this->hemisphereButton = (gcnew System::Windows::Forms::PictureBox());
                this->helpButton = (gcnew System::Windows::Forms::PictureBox());
                this->outputImage = (gcnew System::Windows::Forms::PictureBox());
                this->selectedSurfaceColor = (gcnew System::Windows::Forms::PictureBox());
                this->deleteSurfaceButton = (gcnew System::Windows::Forms::PictureBox());
                this->editSurfaceButton = (gcnew System::Windows::Forms::PictureBox());
                this->surfaceListBox = (gcnew System::Windows::Forms::DataGridView());
                this->surface_type = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->Column6 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->Column7 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->colorPalletImage = (gcnew System::Windows::Forms::PictureBox());
                this->xp0box = (gcnew System::Windows::Forms::TextBox());
                this->yp0box = (gcnew System::Windows::Forms::TextBox());
                this->zp0box = (gcnew System::Windows::Forms::TextBox());
```

EXHIBIT 106

Certificate of Registration

This Certificate issued under the seal of the Copyright Office in accordance with title 17, *United States Code*, attests that registration has been made for the work identified below. The information on this certificate has been made a part of the Copyright Office records.

Acting United States Register of Copyrights and Director

Registration Number
TX 8-400-276
Effective Date of Registration:
January 13, 2017

Title

Title of Work: emoshaGraphics CAD

Completion/Publication

Year of Completion: 2017
Date of 1st Publication: January 10, 2017
Nation of 1st Publication: United States

Author

- **Author:** Louis Arthur Coffelt
Author Created: computer program
Citizen of: United States
Domiciled in: United States
Year Born: 1959

Copyright Claimant

Copyright Claimant: Louis Arthur Coffelt
231 E. Alessandro Blvd., 6A-504, Riverside, CA, 92508, United States

Rights and Permissions

Name: Louis Arthur Coffelt
Email: louis.coffelt@gmail.com
Telephone: (951)790-6086
Address: 231 E. Alessandro Blvd., 6A-504
Riverside, CA 92508 United States

Certification

Name: Louis Arthur Coffelt, Jr.
Date: January 13, 2017
Applicant's Tracking Number: 1133emgcadv1

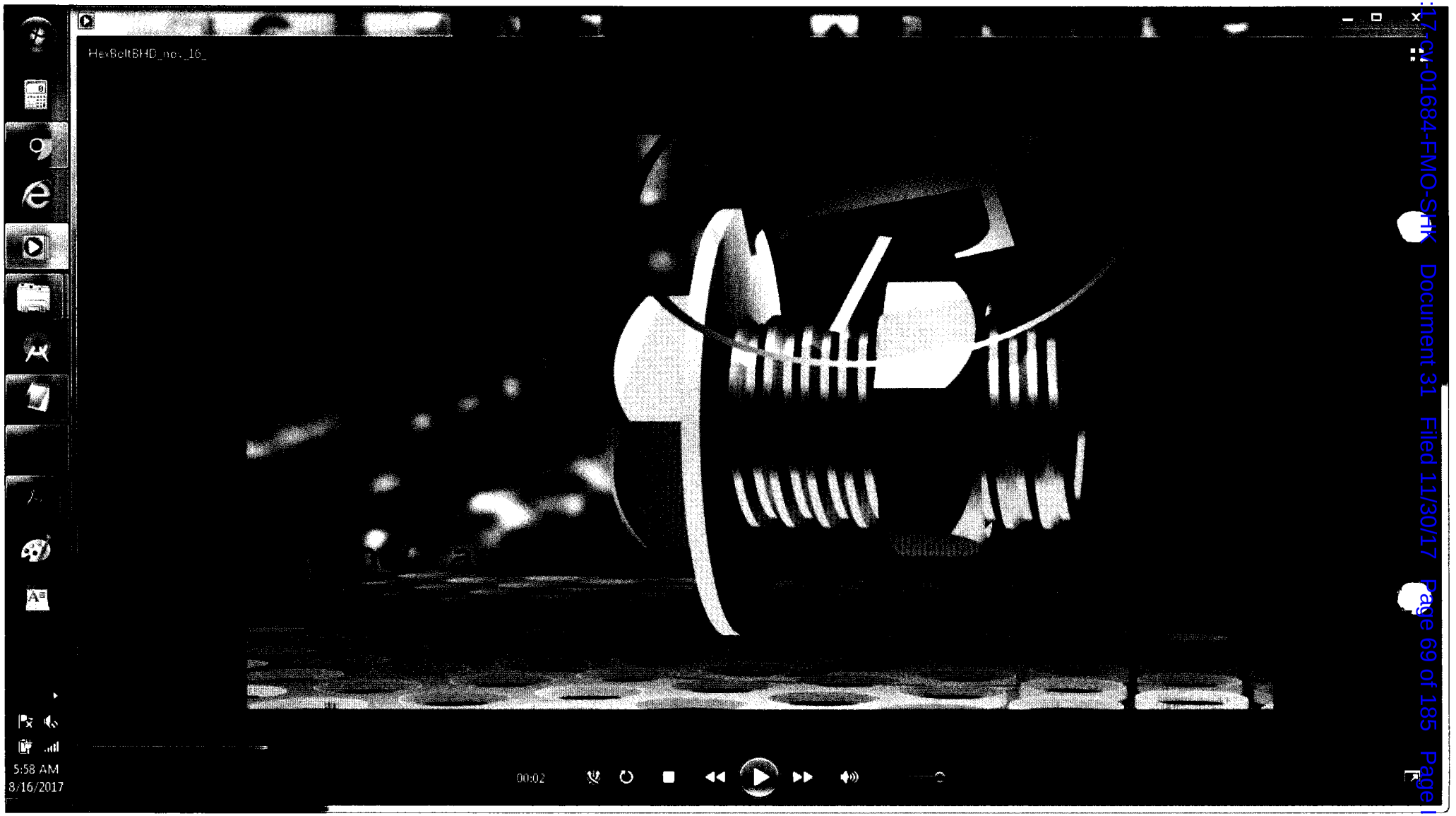
```
#pragma once
#include "C:\\test\\cad_dll\\Convert_Binary_To_Doubles.h"
#include "C:\\test\\cad_dll\\WriteBinaryDoublesOrIntDLL.h"
#include "C:\\test\\cad_dll\\Convert_Doubles_To_Binary.h"


#include "Drawing_Cls.h"
#include <cmath>

namespace CppWinForm1
{

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::IO;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace Convert_Binary_To_Doubles;
	using namespace WriteBinaryDoublesDLLx;
	using namespace Convert_Doubles_To_Binary;
	/// <summary>
	/// Summary for MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{

	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::PictureBox^  fileButton;
	private: System::Windows::Forms::PictureBox^  newOpenCloseSaveButton;

	protected:
```

```
protected:

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;
private: System::Windows::Forms::PictureBox^  viewButton;
private: System::Windows::Forms::PictureBox^  rotateZoomButton;
private: System::Windows::Forms::PictureBox^  coordSystemImage;

private: System::Windows::Forms::PictureBox^  planeButton;
private: System::Windows::Forms::PictureBox^  triangleButton;
private: System::Windows::Forms::PictureBox^  discButton;
private: System::Windows::Forms::PictureBox^  ringButton;

private: System::Windows::Forms::PictureBox^  cylinderButton;

private: System::Windows::Forms::PictureBox^  sphereButton;
private: System::Windows::Forms::PictureBox^  hemisphereButton;



private: System::Windows::Forms::PictureBox^  helpButton;
private: System::Windows::Forms::PictureBox^  outputImage;

private: System::Windows::Forms::PictureBox^  selectedSurfaceColor;
private: System::Windows::Forms::PictureBox^  deleteSurfaceButton;
private: System::Windows::Forms::PictureBox^  editSurfaceButton;
private: System::Windows::Forms::PictureBox^  colorPalletImage;
private: System::Windows::Forms::TextBox^  xp0box;
private: System::Windows::Forms::TextBox^  yp0box;
private: System::Windows::Forms::TextBox^  zp0box;
private: System::Windows::Forms::TextBox^  zp1box;
private: System::Windows::Forms::TextBox^  yp1box;
private: System::Windows::Forms::TextBox^  xp1box;
private: System::Windows::Forms::TextBox^  radiusBox;

private: System::Windows::Forms::PictureBox^  newSurfaceButton;
private: System::Windows::Forms::PictureBox^  saveSurfaceButton;
private: System::Windows::Forms::TextBox^  surfaceDescriptionBox;
private: System::Windows::Forms::PictureBox^  saveDescriptionButton;



private: System::Windows::Forms::PictureBox^  button188x32;

private: System::Windows::Forms::Label^  SelectedColorTipLabel;
private: System::Windows::Forms::Label^  recentFileName0button;
private: System::Windows::Forms::Label^  recentFileName1button;
private: System::Windows::Forms::Label^  recentFileName2button;
private: System::Windows::Forms::Label^  recentFileName3button;
```

```
        private: System::Windows::Forms::TextBox^  zp2box;
        private: System::Windows::Forms::TextBox^  yp2box;
        private: System::Windows::Forms::TextBox^  xp2box;
        private: System::Windows::Forms::TextBox^  zp3box;
        private: System::Windows::Forms::TextBox^  yp3box;
        private: System::Windows::Forms::TextBox^  xp3box;
        private: System::Windows::Forms::PictureBox^  lineButton;
        private: System::Windows::Forms::PictureBox^  filletLinearButton;
        private: System::Windows::Forms::PictureBox^  surfaceDensityButton;
        private: System::Windows::Forms::PictureBox^  lightSourceButton;
        private: System::Windows::Forms::TextBox^  doubleInput2box;
        private: System::Windows::Forms::TextBox^  doubleInput1box;
        private: System::Windows::Forms::TextBox^  doubleInput0box;
        private: System::Windows::Forms::PictureBox^  saveDoublesButton;
        private: System::Windows::Forms::Label^  double0Label;
        private: System::Windows::Forms::Label^  double1Label;
        private: System::Windows::Forms::Label^  double2Label;
        private: System::Windows::Forms::PictureBox^  cancelInputButton;

        private: System::Windows::Forms::PictureBox^  densityChoiceButton;
        private: System::Windows::Forms::Label^  densityValueLabel;
        private: System::Windows::Forms::PictureBox^  startPageButton;
        private: System::Windows::Forms::PictureBox^  start_page_dwg_ico;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  surface_type;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  Column6;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  Column7;
        private: System::Windows::Forms::PictureBox^  cancelSurfaceButton;
        private: System::Windows::Forms::PictureBox^  newProjectButton;

        private: System::Windows::Forms::DataGridView^  surfaceListBox;



#pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                System::ComponentModel::ComponentResourceManager^  resources = (gcnew
System::ComponentModel::ComponentResourceManager(MyForm::typeid));
                System::Windows::Forms::DataGridViewCellStyle^  dataGridViewCellStyle7 = (gcnew
System::Windows::Forms::DataGridViewCellStyle());
                System::Windows::Forms::DataGridViewCellStyle^  dataGridViewCellStyle8 = (gcnew
System::Windows::Forms::DataGridViewCellStyle());
                System::Windows::Forms::DataGridViewCellStyle^  dataGridViewCellStyle9 = (gcnew
System::Windows::Forms::DataGridViewCellStyle());
                this->fileButton = (gcnew System::Windows::Forms::PictureBox());
                this->newOpenCloseSaveButton = (gcnew System::Windows::Forms::PictureBox());
                this->viewButton = (gcnew System::Windows::Forms::PictureBox());
                this->rotateZoomButton = (gcnew System::Windows::Forms::PictureBox());
                this->coordSystemImage = (gcnew System::Windows::Forms::PictureBox());
```

EXHIBIT 107

Computer File Location: Coffelt's Laptop, Service Tag: GMBTY32



69

COFFELT'S RESULTS OF VECTOR WORK, GRADIENT WORK, STERADIAN WORK

EXHIBIT 108

Computer File Location: Coffelt's Laptop, Service Tag: GMBTY32

COFFELT'S RESULTS CAD WORK

71

EXHIBIT 109

Computer File Location: Coffelt's Laptop, Service Tag: GMBTY32

COFFELT'S RESULTS CAD WORK

EXHIBIT 110

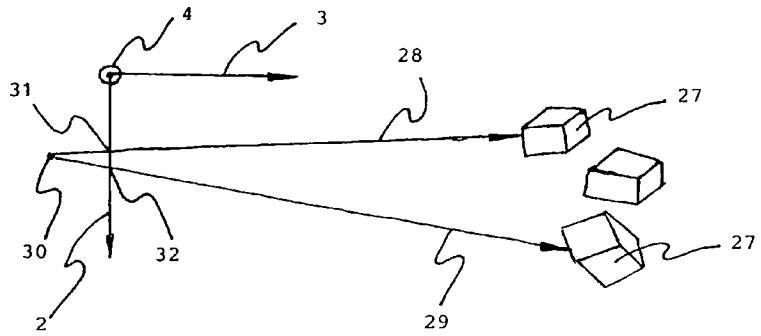


FIG. 7

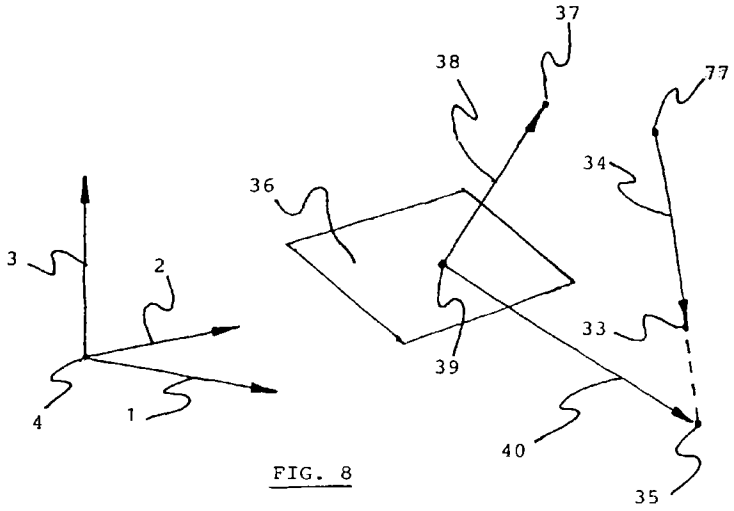


FIG. 8

COFFELT'S VECTOR WORK

columnD−pti*pixelsPerInchD; column−unsigned int(columnD), Indexx=row*bitmapPixelWidth+column; priorlength=lengthv[Indexx]; currentlength=sqrt (vppti*vppti+vpptj*vpptj+vpptk*vpptk); if(currentlength<priorlength) <[lengthv[Indexx]=currentlength: |> pti+=0.001; count1++; |> while(count2<700) <[ptj=m0*pti−4.3; vppti=vpi−pti; vpptj=vpj−ptj; vpptk=vpk−ptk; IntersectVectorWithPlane(intpti, intptj, intptk, vpi, vpj, vpk, pti, ptj, ptk, N1*i*, N1*j*, N1*k*, N0*i*, N0*j*, N0*k*); if(intpti<0.0 or intpti> bitmapWidthInches or intptj<0.0 or intptj> bitmapHeightInches)<[pti+=0.001; count2++; if(count2<700)<|continue; |> else <|break; |>|> rowD=ptj*pixelsPerInchD; row=unsigned int(rowD); columnD=pti* pixelsPerInchD; column=unsigned int(columnD); Indexx=row* bitmapPixelWidth+column; priorlength=lengthv[Indexx]; currentlength=sqrt (vppti*vppti+vpptj*vpptj+vpptk*vpptk); difference1=abs (currentlength−priorlength); if(difference1<0.0001)< [pointIsVisible=true; red=240; green=0; blue=0; bitmapx.SetPixel(row, column, red, green, blue); |> else <[pointIsVisible=false; |> pti+=0.001; count2++; |> while (count3<900)<[ptj=m1*pti−2.89; vppti=vpi−pti; vpptj=vpj−ptj; vpptk=vpk−ptk; IntersectVectorWithPlane(intpti, intptj, intptk, vpi, vpj, vpk, pti, ptj, ptk, N1*i*, N1*j*, N1*k*, N0*i*, N0*j*, N0*k*); if(intpti<0.0 or intpti> bitmapWidthInches or intptj<0.0 or intptj> bitmapHeightInches)<[pti+=0.001; count3++; if(count3<900)<|continue; |> else<|break; |>|> rowD−ptj*pixelsPerInchD; row−unsigned int(rowD); columnD=pti*pixelsPerInchD; column=unsigned int(columnD); Indexx−row*bitmapPixelWidth+column; priorlength−lengthv[Indexx]; currentlength=sqrt (vppti*vppti+vpptj*vpptj+vpptk*vpptk); difference1−abs (currentlength−priorlength); if(difference1<0.0001)< [pointIsVisible=true; red−0; green−0; blue=250; bitmapx.SetPixel(row, column, red, green, blue); |> else <[pointIsVisible−false; |> pti+−0.001; count3++; |>|>

FIG. **8** shows a perspective view of a vector (**34**) intersecting a plane (**36**) at point (**35**). A normal vector (**38**) of plane (**36**) is shown. The normal vector is formed by point N0 (**39**) and point N1 (**37**). Point N (**39**) is on plane (**36**). Vector (**34**) is formed of any two points, point (**77**) and point (**33**). Vector (**40**) is formed of two points, point (**35**) and point (**39**). Vector (**40**) is in plane (**36**). The following c++ code shows an example of 'Vector Plane Intersection', and is the function used above, where, intpti, intptj, intptk is intersection point (**35**); pt1 is point (**77**). pt0 is point (**33**); N0 is point (**39**); N1 is point (**37**).

c++ <|void IntersectVectorWithPlane(double& intptiP, double& intptjP, double& intptkP, double pt1*i*, double pt1*j*, double pt1*k*, double pt0*i*, double pt0*j*, double pt0*k*, double N1*i*P, double N1*j*P, double N1*k*P, double N0*i*P, double N0*j*P, double N0*k*P)<|double Ni−N1*i*P−N0*i*P; double Nj−N1*j*P−N0*j*P; double Nk N1*k*P−N0*k*P; double testdenom−abs(pt1*i*−pt0*i*); if(testdenom<1.0e−9)<|return; |> double mji−(pt1*j*−pt0*j*)/(pt1*i*−pt0*i*); double mki (pt1*k*−pt0*k*)/(pt1*i*−pt0*i*); testdenom−abs(Ni+Nj*mji+Nk*mki); if(testdenom<1.0e−9) <|return; |> tempi−(N0*i*P*Ni+Nj*mji*pt0*i*−Nj*pt0*j*+Nj*N0*j*P+Nk*mki*pt0*i*−Nk*pt0*k*+Nk*N0*k*P)/(Ni+Nj*mji+Nk*mki); intptiP−tempi; intptjP−mji*(tempi−pt0*i*)+pt0*j*; intptkP mki*(tempi−pt0*i*)+pt0*k*; |>|>

The general formula for intptiP is formed by combining a general formula for a plane with a general formula for a line. More specifically, the projection of a vector to the i-j plane, and the projection of the vector to the i-k plane. The general formula for a plane is set in an equation that any vector in the plane dotted with the planes normal vector is zero. N dot v is zero, where N is the normal vector of the plane, and v is any vector in the plane. A general formula for a line is. j−mji*(i−i0)+j0 where i0, j0, and mji are given. Also, k−mki*(i−i0)+k0 where i0, k0, and mki are given. These two equations for a line are substituted in the equation for a plane; and solved for i. This substitution eliminates variables j and k. Only i remains in the equation. Solving for i yields the equation above in the c++code snippet.

The graphic object structure analysis may also include reflection vectors. A reflection vector can be derived for any point in a graphic object. This reflection vector may intersect any graphic object. e.g. any plane, sphere, or surface. The intersection point can be assigned any selected pixel color. For example, a light source contacts a particular point on a blue surface; a reflection vector is calculated at this particular point; the reflection vector intersects a red sphere; the intersection point is set to blue. The following c++code snippet sets forth an example to calculate a reflection vector.

c++ <|void ReflectionVector(double& rpti, double & rptj, double& rptk, double Ni, double Nj, double Nk, double s1*i*, double s1*j*, double s1*k*, double ai, double aj, double ak)< |double si=s1*i*−ai; double sj−s1*j*−aj; double sk=s1*k*−ak; double lengths=sqrt(si*si+sj*sj+sk*sk); double lengthN−sqrt(Ni*Ni+Nj*Nj+Nk*Nk); double NdotS= (Ni*si+Nj*sj+Nk*sk)/(lengthN*lengths); double testDot=abs(NdotS); if(testDot<1.0e−6)<|return; |> else if(testDot>0.9999)<|rpti=s1*i*; rptj−s1*j*; rptk−s1*k*; return; |> double phi=a cos(NdotS); double Nri=0.0; double Nrj−0.0; double Nrk−0.0; double rxi=0.0; double rxj−0.0; double rxk=0.0; double tlen−2.0*lengths*sin(phi); Nri=Nj*sk−sj*Nk; Nrj=−(Ni*sk−si*Nk); Nrk=Ni*sj−si*Nj; rxi−Nj*Nrk−Nrj*Nk; rxj−−(Ni*Nrk−Nri*Nk); rxk=Ni*Nrj−Nri*Nj; double lengthrx=sqrt(rxi*rxi+rxj*rxj+ rxk*rxk); if(lengthrx>1.0e−6)<|cx=tlen/lengthrx; |> else <|return; |> double tri=cx*rxi; double trj−cx*rxj; double trk−cx*rxk; rpti=s1*i*+tri; rptj=s1*j*+trj; rtpk=s1*k*+trk; |>|>

In the above example for the reflection vector. The tail end of the reflection vector is ai, aj, ak; the terminal end of the reflection vector is rpti, rptj, rptk. The reflection vector is: (rpti−ai)i+(rptj−aj)j+(rptk−ak)k: ai, aj, ak. the intersection point of a light source vector with the plane. Ni, Nj, Nk is the normal vector of the plane.

A graphic object structure may also include translucent surfaces. A translucent surface can be derived by intersecting a vector with one or more surfaces. Next, set the selected pixel color at a relatively less pixel per inch resolution. For example, for a bitmap having a resolution of 1000 pixels per inch, a translucent surface can be attained by setting a background image at about 500 pixels per inch. For example, a foreground rectangular translucent blue surface; and a background linear red surface; initially all pixels in the bitmat are blue: Next, the program iterates thru the equation of the line; and assign a red pixel at a density of 500 pixels per inch.

One method to create a translucent surface is using a 'next least length' concept. This concept is related to a 'visible' point on the geometric object. Opaque surfaces described above, have only one point per steradian which is 'visible'. In comparison, for a translucent surface, there may be to or three or more points in one particular steradian which are 'visible'. e.g. the visible points are 2 points having the least position vector length. The following is an example c++ code snippet showing a method to calculate a translucent surface:

The following is a summary of c++ code for translucent surface calculation: This code may be set in line with the above c++ code examples; c++ <vector> priorLength0, <vector> priorLength1, and <vector> priorLength2 contain values having a respective next least length: for example, at index **33**. priorLength0[**33**]—22.15; priorLength1[**33**]—28.76; prior-

EXHIBIT 111

Ambient Shadow in Sho | Microsoft Word - Comp | AutoCAD 2009 and Auto

Secure https://books.google.com/books?id=NWVe6iB8IcEC&pg=PA765&lpg=PA765&dq=autocad+2009+shadows&source=bl&ots=i7cWKI3ZDg&sig=nD5tz7tqR1xvX68N0S

Apps | Inbox (104) - louis.co | Google Maps | (1) Facebook | YouTube | The Largest 4G LTE N | Timeless - Live (With | Coldplay-Things I Do | tryad - beauty - YouT | Other bookmark

Google autocad 2009 shadows

Books

Add to my library Write review

Page 771

Result 4 of 10 in this book for **autocad 2009 shadows** - ‹ Previous Next › - View all

Clear search

VIEW EBOOK

Get this book in print ▼

G+1 0

0 Reviews
Write review

AutoCAD 2009 and AutoCAD LT 2009: No Experience Required

By Jon McFarland

autocad 2009 shadows Go

About this book

- My library
- My History

Books on Google Play

WILEY

Pages displayed by permission of John Wiley & Sons. Copyright.

box should look similar to Figure 17.17. The background image not only appears behind the cabin and ground, but it also contributes light to the scene.

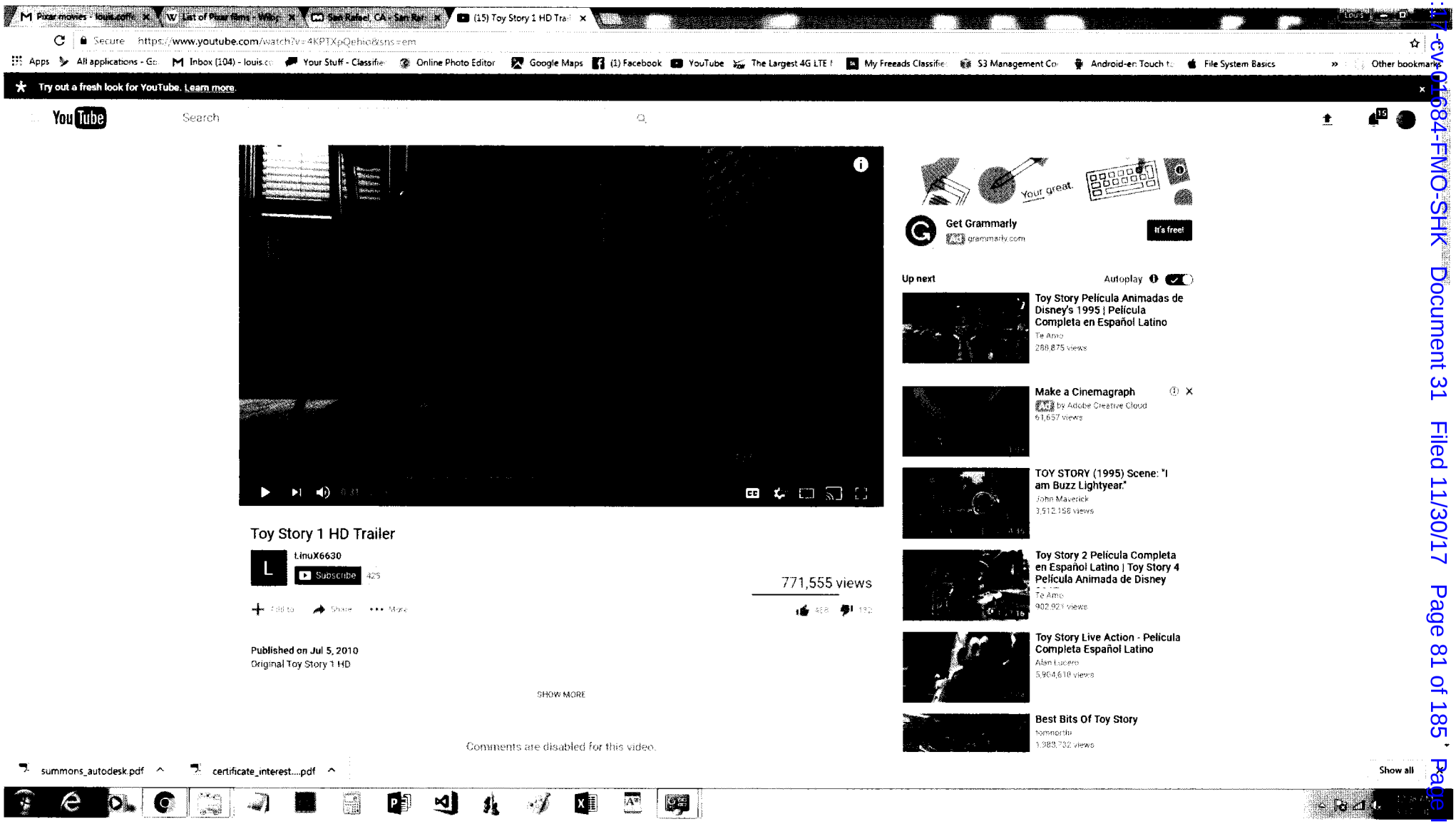
FIGURE 17.17: The cabin rendered with the Sun & Sky background and additional illumination

9. Save your drawing as `Cabin17b.dwg`.

State of Art Surface Gradient Years 1970 through 2010

State of Art Surface Gradient Years 1970 through 2010

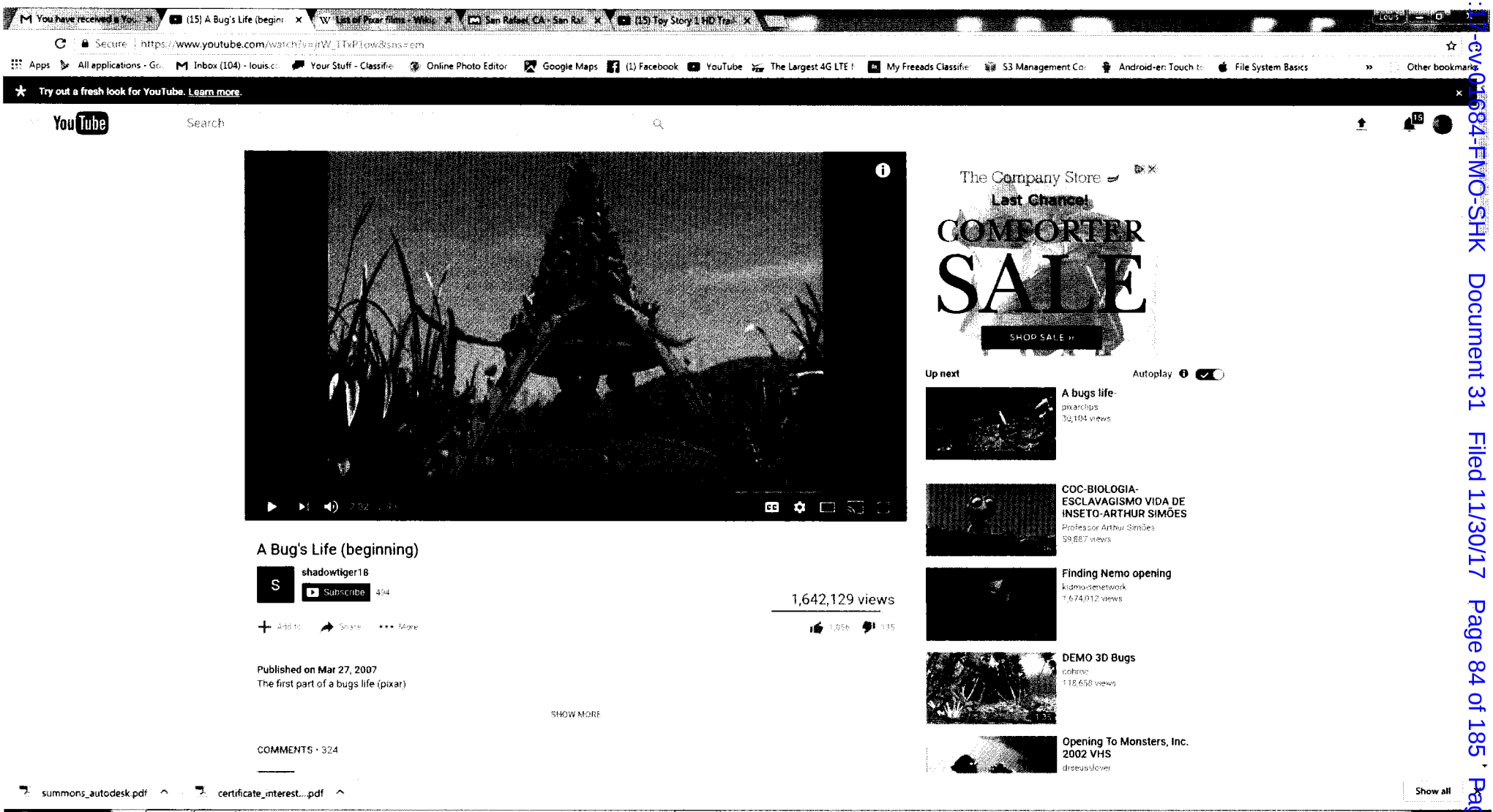
EXHIBIT 112



PIXAR RESULTS 1995

PIXAR RESULTS 1995

EXHIBIT 113

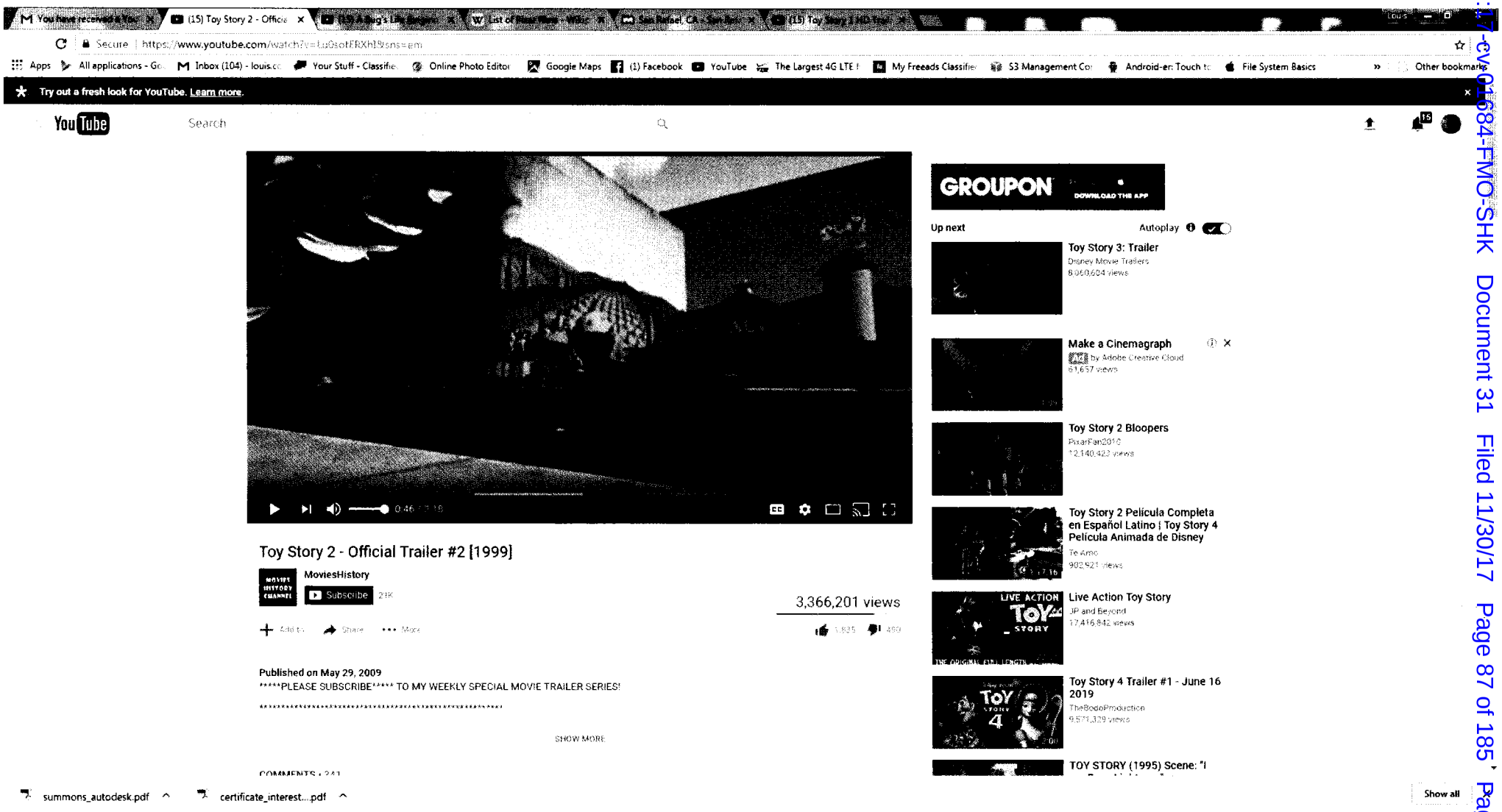


PIXAR RESULTS 1998

48

PIXAR RESULTS 1998

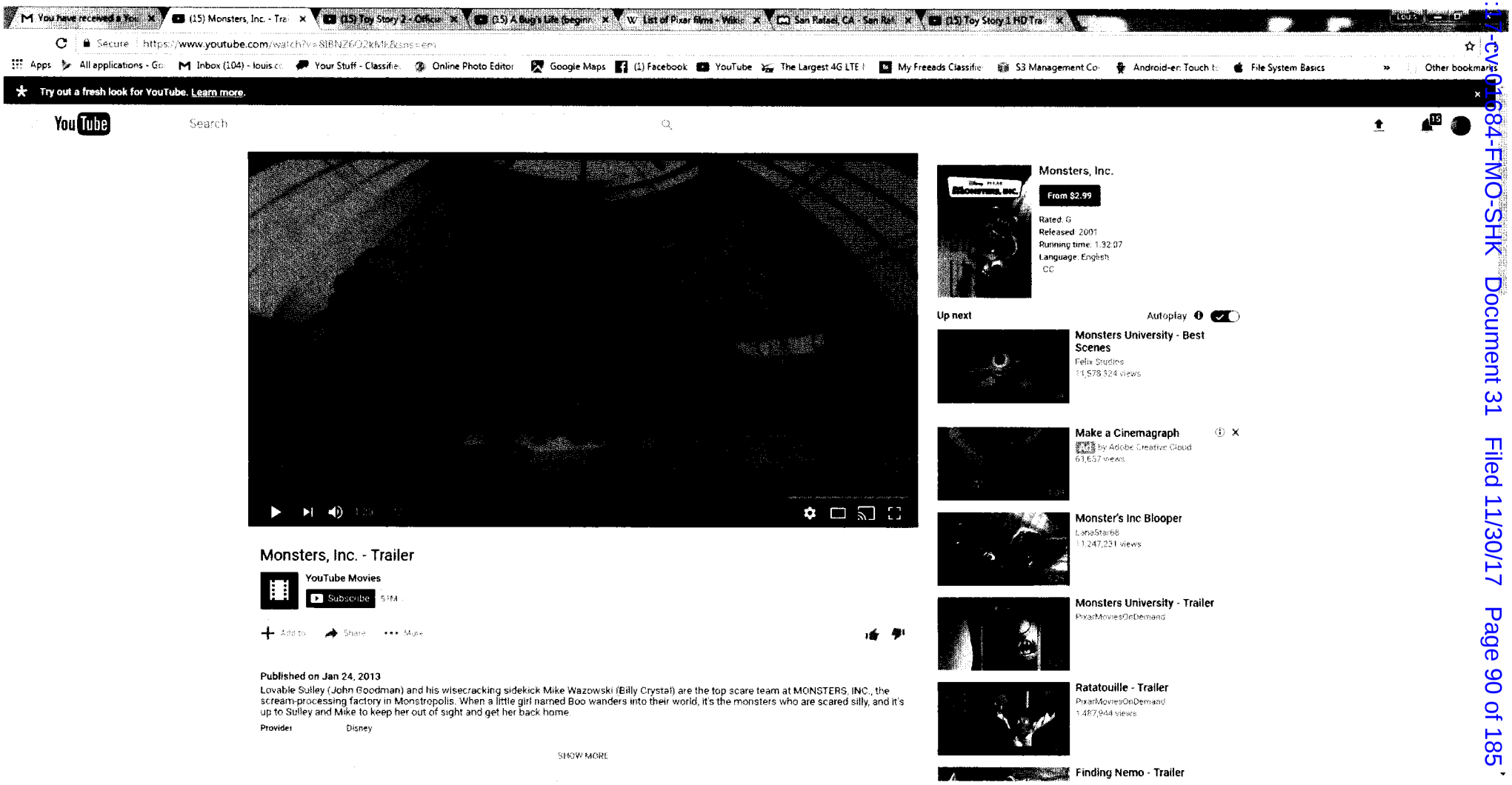
EXHIBIT 114



PIXAR RESULTS 1999

PIXAR RESULTS 1999

EXHIBIT 115

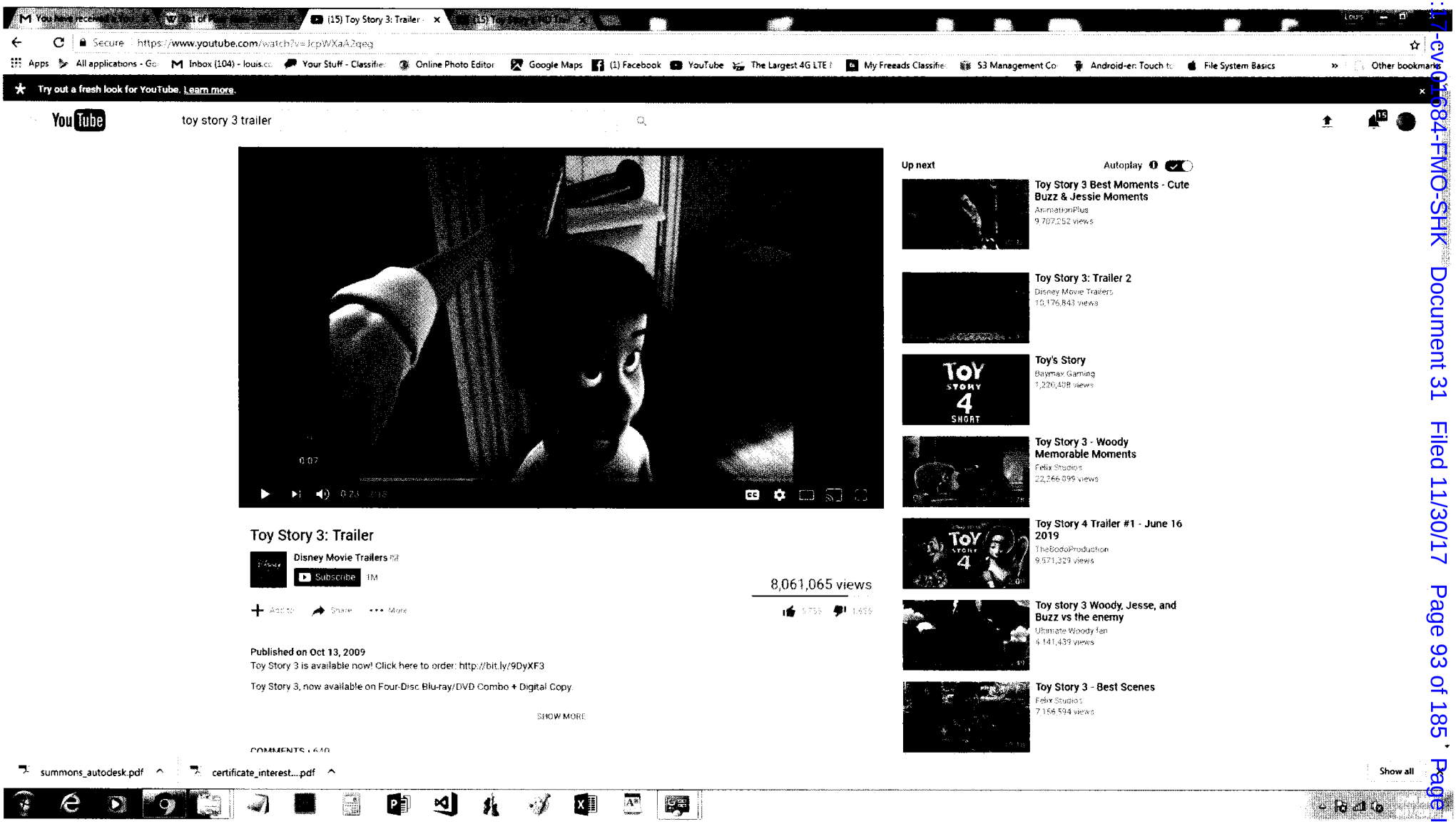


PIXAR RESULTS 2001

90

PIXAR RESULTS 2021

EXHIBIT 116

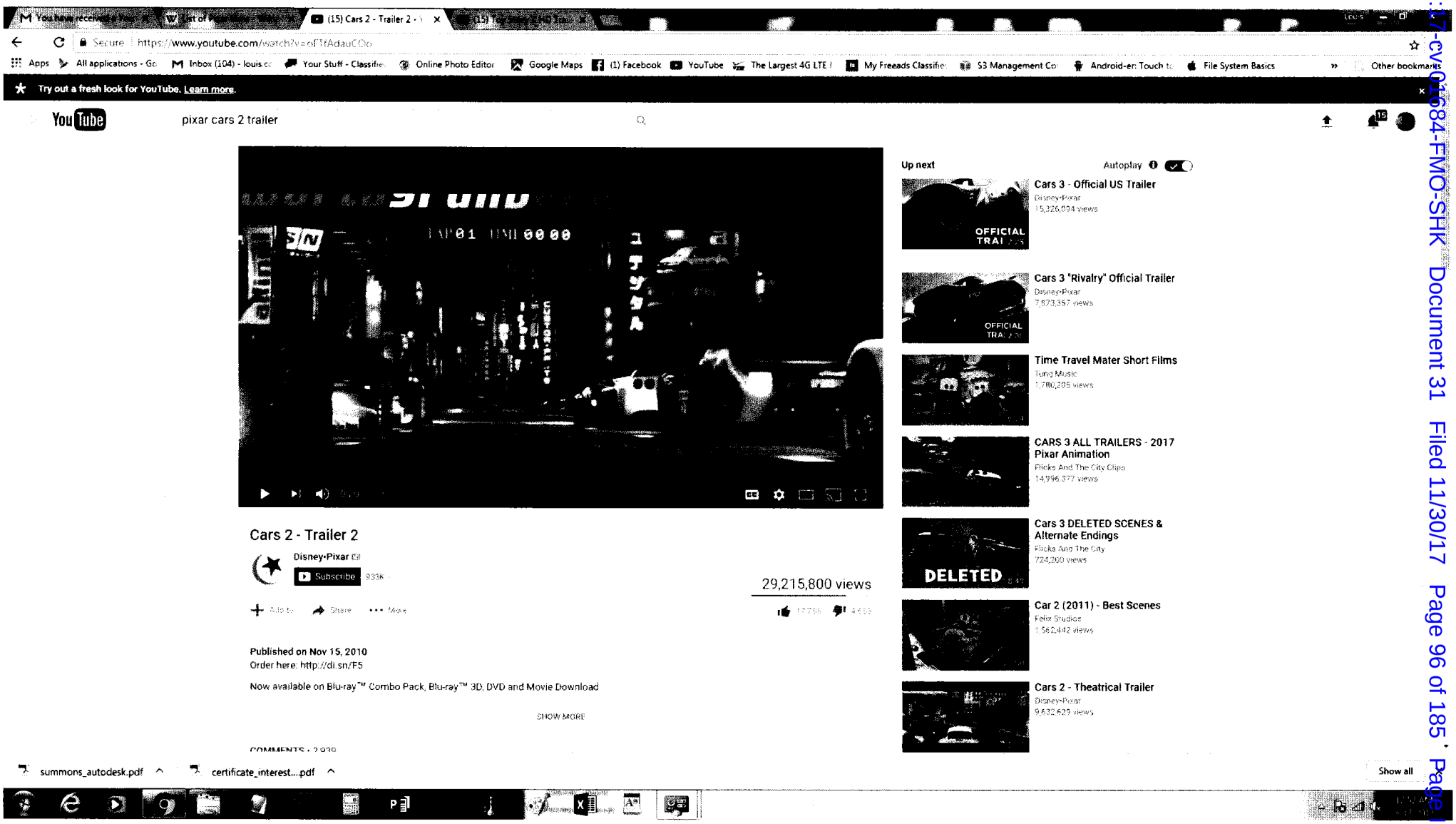


PIXAR RESULTS 2010

36

PIXAR RESULTS 2010

EXHIBIT 117

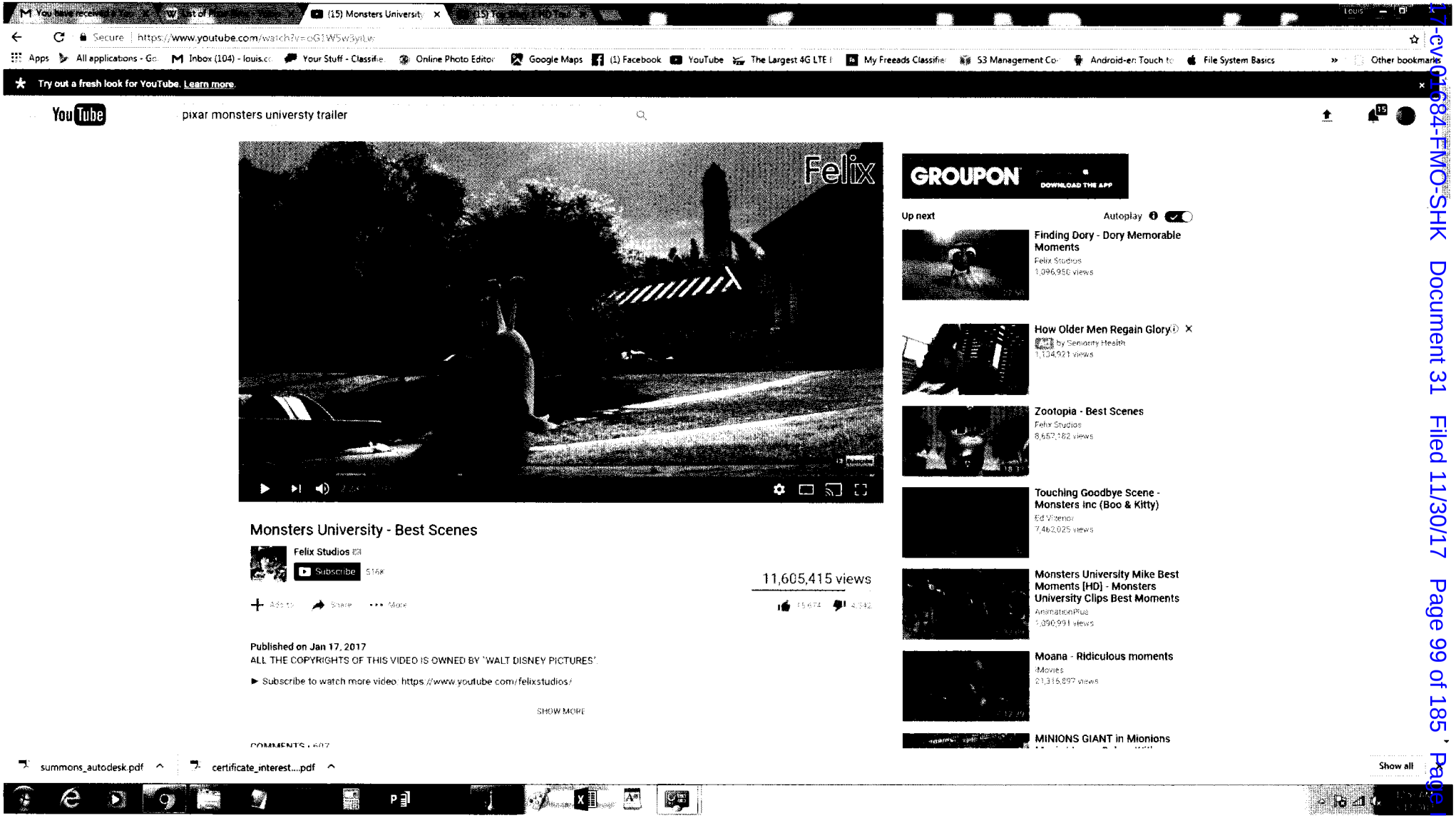


PIXAR RESULTS 2011

96

PIXAR RESULTS 2011

EXHIBIT 118

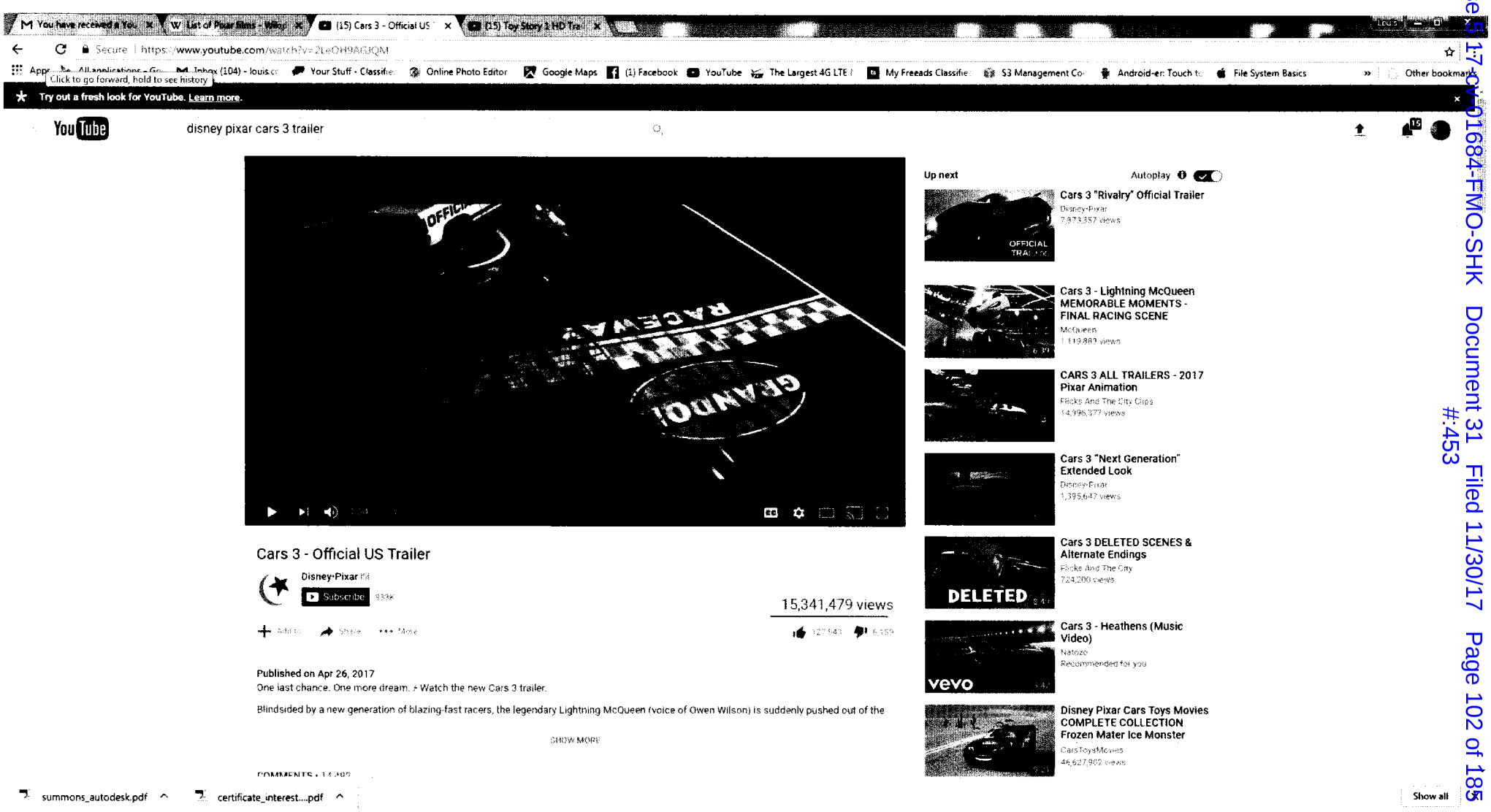


99

PIXAR RESULTS 2013

PIXAR RESULTS 2013

EXHIBIT 119

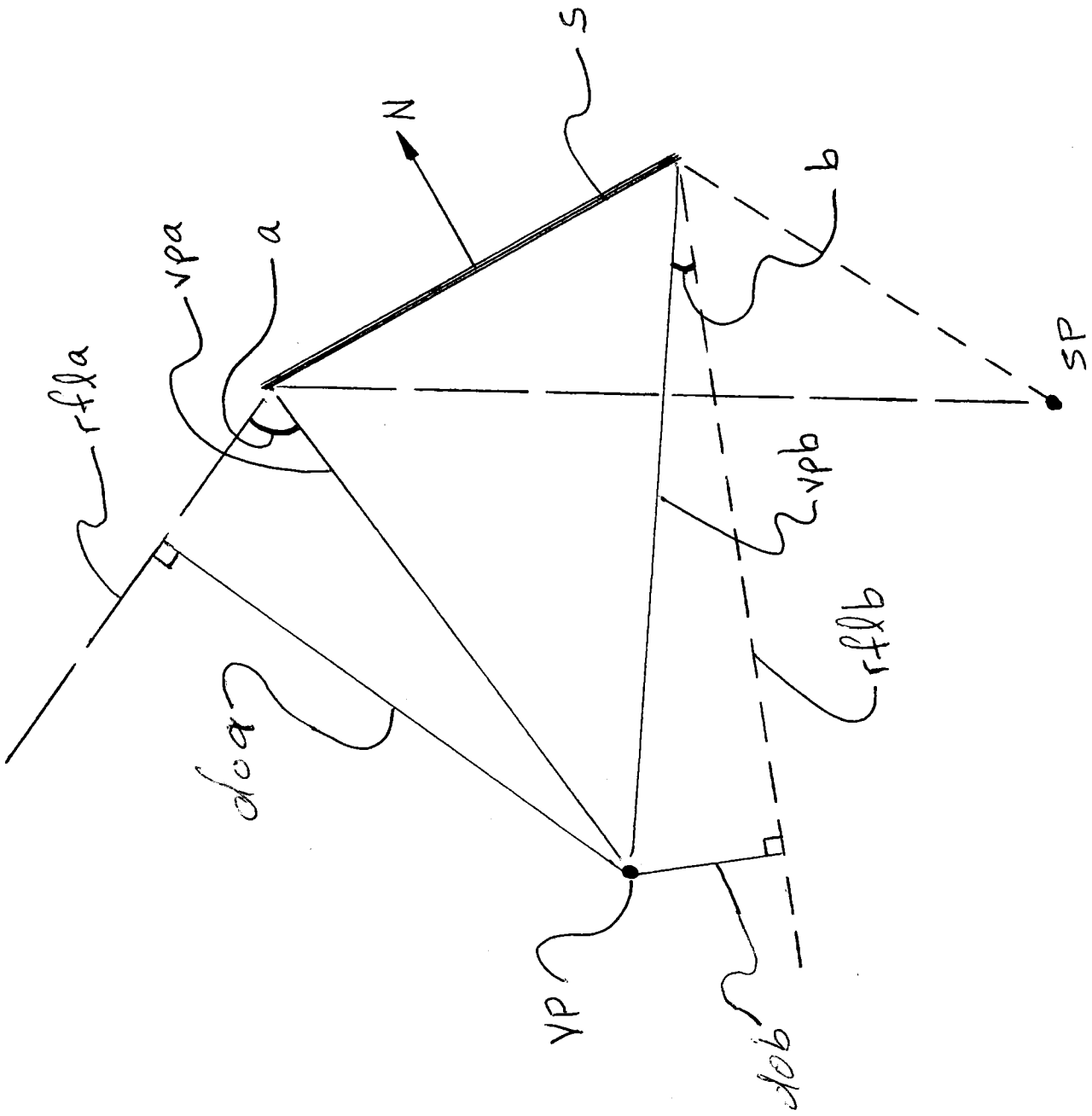


PIXAR RESULTS 2017

201

PIXAR RESULTS 2017

EXHIBIT 120



COFFELT GRADIENT WORK

$$mgrad = \frac{-50}{(maxd - mind)}$$

color shift = mgrad (do - mind)

COFFELT GRADIENT WORK

EXHIBIT 121

STATEMENT OF FACTS RE:
OSL COPYRIGHT INFRINGEMENT

The following OSL computer program is an infringement of Coffelt's Gradient Work 2010, U.S. application No. 1-5121154211.
The following OSL computer program is an infringement of Coffelt's Photorealistic Gradient Work, U.S. application No. 1-5376971191.
The following OSL computer program is an infringement of Coffelt's Gradient Work, Registration No. TXu002049564.

The location of OSL infringing source code is identified with reference (A) through (J) in this exhibit.
The basis for OSL is a derivative of Coffelt's source code (Complaint at paragraph 66) is in the last pages of this exhibit.

On Tuesday, August 01, 2017, 6:16:42 PM Coffelt downloaded a copy of OSL source code from:

https://github.com/imageworks/OpenShadingLanguage **(Infringing URL)**

The OSL Infringing Code is as follows:

OSL file name:

OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp at line 174:

(A)

```
virtual float sample(const OSL::ShaderGlobals& sg, float rx, float ry, float rz, OSL::Dual2<OSL::Vec3>& wi, float& pdf) const
    {
        float cosNO = -N.dot(sg.I);          (B)
        if (cosNO > 0)
        {
            // reflect the view vector
            Vec3 R = (2 * cosNO) * N + sg.I;
            TangentFrame tf(R);
            float phi = 2 * float(M_PI) * rx;          (C)
            float sp, cp;
            OIIO::fast_sincos(phi, &sp, &cp);
            float cosTheta = OIIO::fast_safe_pow(ry, 1 / (exponent + 1));
            float sinTheta2 = 1 - cosTheta * cosTheta;
            float sinTheta = sinTheta2 > 0 ? sqrtf(sinTheta2) : 0;
            wi = tf.get(cp * sinTheta, sp * sinTheta, cosTheta);
            // leave derivs 0?
            float cosNI = N.dot(wi.val());
            if (cosNI > 0)
        {
        pdf = (exponent + 1) * float(M_1_PI / 2) * OIIO::fast_safe_pow(cosTheta, exponent);
        return cosNI * (exponent + 2) / (exponent + 1);
    }
}
return pdf = 0;
```

109

OSL file name:

OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp at line 550:

```
Vec3 sampleMicronormal(const Vec3 wo, float randu, float randv) const
{
    /* Project wo and stretch by alpha values */
    Vec3 swo = wo;
    swo.x *= xalpha;
    swo.y *= yalpha;
    swo = swo.normalize();
    // figure out angles for the incoming vector
    float cos_theta = std::max(swo.z, 0.0f);
    float cos_phi = 1;
    float sin_phi = 0;
    /* Normal incidence special case gets phi 0 */
    if (cos_theta < 0.99999f)
    {
        float invnorm = 1 / sqrtf(SQR(swo.x) + SQR(swo.y));
        cos_phi = swo.x * invnorm;
        sin_phi = swo.y * invnorm;
    }
    Vec2 slope = Distribution::sampleSlope(cos_theta, randu, randv);
    /* Rotate and unstretch slopes */
    Vec2 s(cos_phi * slope.x - sin_phi * slope.y, sin_phi * slope.x + cos_phi * slope.y);
    s.x *= xalpha;
    s.y *= yalpha;
    float mlen = sqrtf(s.x * s.x + s.y * s.y + 1);
    Vec3 m(fabsf(s.x) < mlen ? -s.x / mlen : 1.0f, fabsf(s.y) < mlen ? -s.y / mlen : 1.0f, 1.0f / mlen);
    return m;
```

(E)

(D)

110

OSL file name:

OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.h at line 50:

```
Color3 sample(const ShaderGlobals& sg, float rx, float ry, float rz, Dual2<Vec3>& wi, float& pdf) const
{
        float accum = 0;
        for (int i = 0; i < num_bsdfs; i++)
        {
                if (rx < (pdfs[i] + accum))
        {
        rx = (rx - accum) / pdfs[i];

        rx = std::min(rx, 0.99999994f);
        // keep result in [0,1)
        Color3 result = weights[i] * (bsdfs[i]->sample(sg, rx, ry, rz, wi, pdf) / pdfs[i]);
        pdf *= pdfs[i];
        // we sampled PDF i, now figure out how much the other bsdfs contribute to the chosen direction
        for (int j = 0; j < num_bsdfs; j++)
        {
                if (i == j) continue;
                float bsdf_pdf = 0;
                Color3 bsdf_weight = weights[j] * bsdfs[j]->eval(sg, wi.val(), bsdf_pdf);
                MIS::update_eval(&result, &pdf, bsdf_weight, bsdf_pdf, pdfs[j]);
        }
        return result;
}
accum += pdfs[i];
```

(F)

OSL file name:

OpenShadingLanguage-master\src\testrender\testrender.cpp at line 418

```
Color3 subpixel_radiance(float x, float y, Sampler& sampler, ShadingContext* ctx)
{
        Ray r = camera.get(x, y);
        Color3 path_weight(1, 1, 1);

        Color3 path_radiance(0, 0, 0);
        int prev_id = -1;
        float bsdf_pdf = std::numeric_limits<float>::infinity();
        // camera ray has only one possible direction
        bool flip = false;
        for (int b = 0; b <= max_bounces; b++)
        {
                // trace the ray against the scene
                Dual2<float> t; int id = prev_id;
                if (!scene.intersect(r, t, id))
        {
        // we hit nothing? check background shader
        if (backgroundShaderID >= 0)
{
if (backgroundResolution > 0)
{
        float bg_pdf = 0;

        Vec3 bg = background.eval(r.d.val(), bg_pdf);

        path_radiance += path_weight * bg * MIS::power_heuristic<MIS::WEIGHT_WEIGHT>(bsdf_pdf, bg_pdf);
}
```

(I) (G) (H) (J)

2 1 1

FACTS AND BASIS OSL IS A DERIVATIVE OF COFFELT'S WORK

(A)

Coffelt: 0000 rflx = rptx - ptx00a;
0001 rfly = rpty - pty00a;
0002 rflz = rptz - ptz00a;

OSL: OSL::Dual2<OSL::Vec3>& wi

Basis: (i) Coffelt's source code ("rflx, rfly, rflz") in the Complaint at paragraph 66 lines 0000 through 0002 are components of the reflection vector based on the surface normal N and the view direction.

(ii) ("wi") is type ("Vec3") and therefore is a vector having 3 components x, y, and z (c++ standard variable).

(iii) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(iv) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(v) The location of this OSL source code ("OSL::Dual2<OSL::Vec3>& wi") is file name: ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line: 175.

(vi) This OSL source code ("OSL::Dual2<OSL::Vec3>& wi") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, This OSL source code ("OSL::Dual2<OSL::Vec3>& wi") is equivalent to Coffelt's source code in the Complaint at paragraph 66 lines 0000 through 0002.

(B)

0003 lenrfl = sqrt(rflx * rflx + rfly * rfly + rflz * rflz);

Coffelt: 0004 vpdotrfl = (vpax * rflx + vpay * rfly + vpaz * rflz) / (lenvpa * lenrfl);

OSL: float cosNO = -N.dot(sg.I);

Basis: (i) dot product is the cosine of angle between 2 vectors See EXHIBIT 124.

(ii) Coffelt's source code ("vpdotrfl") in the Complaint at paragraph 66, line 0004 is the dot product of the view vector and reflection vector. ("lenrfl") the length of the reflection vector; is a standard component of a vector dot product equation.

(iii) ("sg.I") inherently contains the length of a vector. The length of a vector is a required component of a vector dot product. See EXHIBIT 124 ("magnitude of the vector").

(iv) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(v) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms OSL source code ("float cosNO = -N.dot(sg.I)") is directed to the dot product between the reflection vector and direction of view.

(vi) The location of this OSL source code ("float cosNO = -N.dot(sg.I)") is file name: ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line: 176.

(vii) This OSL source code ("float cosNO = -N.dot(sg.I)") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code ("float cosNO = -N.dot(sg.I)") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0003 and line 0004.

(C)

Coffelt: 0005 theta = acos(vpdotrfl);

OSL: float phi = 2 * float(M_PI) * rx;

Basis: (i) Coffelt's source code ("theta") in the Complaint at paragraph 66, line 0005 is the angle between the view vector and reflection vector.

(ii) OSL source code ("float phi = 2 * float(M_PI) * rx") is the angle between the view vector and reflection vector.

(iii) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(iv) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(v) The location of this OSL source code ("float phi = 2 * float(M_PI) * rx") is file name: ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line: 181.

(vi) This OSL source code ("float phi = 2 * float(M_PI) * rx") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code ("float phi = 2 * float(M_PI) * rx") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0005.

511

(D)

Coffelt: 0006 mgrad = -50 / (max_d - min_d);

OSL: Vec2 slope = Distribution::sampleSlope(cos_theta, randu, randv);

Basis: (i) Coffelt's source code ("mgrad ") in the Complaint at paragraph 66, line 0006 is the slope of a line between the maximum distance and the minimum distance to the view point. *See* EXHIBIT 120.

(ii) OSL source code (" Vec2 slope = Distribution::sampleSlope(cos_theta, randu, randv)") is the slope of a line between the maximum distance and the minimum distance to the view point.

(iii) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(iv) The term ("Distribution::sampleSlope") confirms that the slope is based on the distribution containing the maximum distance and the minimum distance to the view point. e.g. the equation uses ("cos_theta") the angle between the direction of view and the reflection vector.

(v) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(vi) The location of this OSL source code ("Vec2 slope = Distribution::sampleSlope(cos_theta, randu, randv)") is file name:
("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line: 569.

(vii) This OSL source code (" Vec2 slope = Distribution::sampleSlope(cos_theta, randu, randv)") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code (" Vec2 slope = Distribution::sampleSlope(cos_theta, randu, randv)") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0006.

(E)

Coffelt: 0006 mgrad = -50 / (max_d - min_d);

OSL: float cos_theta = std::max(swo.z, 0.0f);

Basis: (i) Coffelt's source code ("max_d ") in the Complaint at paragraph 66, line 0006 is the maximum distance between the view vector and reflection vector.

(ii) OSL source code ("float cos_theta = std::max(swo.z, 0.0f)") is the maximum distance between the view vector and reflection vector.

(iii) ("std::max") is a standard C++ function which derives a maximum value of 2 values. *See* EXHIBIT 125.

(iv) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(v) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(vi) The location of this OSL source code ("float cos_theta = std::max(swo.z, 0.0f)") is file name: ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line: 558.

(vii) This OSL source code (" float cos_theta = std::max(swo.z, 0.0f)") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code (" float cos_theta = std::max(swo.z, 0.0f)") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0006.

117

(F)

Coffelt: 0006 mgrad = -50 / (max_d - min_d);

OSL: rx = std::min(rx, 0.99999994f);

Basis: (i) Coffelt's source code ("min_d ") in the Complaint at paragraph 66, line 0006 is the minimum distance between the view vector and reflection vector.

(ii) OSL source code ("rx = std::min(rx, 0.99999994f)") is the minimum distance between the view vector and reflection vector.

(iii) ("std::min") is a standard C++ function which derives a minimum value of 2 values. *See* EXHIBIT 126.

(iv) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(v) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(vi) The location of this OSL source code ("rx = std::min(rx, 0.99999994f)") is file name: ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.h") at line: 55.

(vii) This OSL source code (" rx = std::min(rx, 0.99999994f)") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code ("rx = std::min(rx, 0.99999994f)") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0006.

811

(G)

Coffelt: 0007 d0 = lenvpa * sin(theta);

OSL: Vec3 bg = background.eval(r.d.val(), bg_pdf);

Basis: (i) Coffelt's source code ("d0 ") in the Complaint at paragraph 66, line 0007 is the distance from the view point to the reflection vector.

(ii) OSL source code ("r.d") is is the distance from the view point to the reflection vector.

(iii) The parameter ("<MIS::WEIGHT_WEIGHT>") in file name: ("OpenShadingLanguage-master\src\testrender\testrender.cpp") at line: 434, confirms ("r.d") is the distance from the view point to the reflection vector.

(iv) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(v) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(vi) The location of this OSL source code ("r.d") is file name: ("OpenShadingLanguage-master\src\testrender\testrender.cpp") at line: 433.

(vii) This OSL source code ("r.d") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code ("r.d") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0007.

(H)

Coffelt: 0008 shiftd = mgrad * (d0 - min_d);

OSL: path_weight * bg

Basis: (i) Coffelt's source code ("shiftd ") in the Complaint at paragraph 66, line 0008 is the numerical quantity of addition to the base color of the surface.

(ii) OSL source code ("path_weight * bg") is the numerical quantity of addition to the base color of the surface.

(iii) The parameter ("<MIS::WEIGHT_WEIGHT>") in file name: ("OpenShadingLanguage-master\src\testrender\testrender.cpp") at line: 434, confirms ("path_weight * bg") is the numerical quantity of addition to the base color of the surface.

(iv) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(v) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(vi) The location of this OSL source code ("path_weight * bg") is file name: ("OpenShadingLanguage-master\src\testrender\testrender.cpp") at line: 434.

(vii) This OSL source code ("path_weight * bg") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code ("path_weight * bg") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0008.

(I)

Coffelt: 0009 blueD = 100.0;
0010 greenD = 255.0;
0011 redD = 100.0;

OSL: Color3 path_radiance(0, 0, 0);

Basis: (i) Coffelt's source code ("blueD, greenD, redD ") in the Complaint at paragraph 66, line 0009 through line 0011 is declaration of the initial surface color integers.

(ii) OSL source code ("Color3 path_radiance(0, 0, 0)") is declaration of the initial surface color integers.

(iii) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(iv) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(v) The location of this OSL source code ("Color3 path_radiance(0, 0, 0)") is file name: ("OpenShadingLanguage-master\src\testrender\testrender.cpp") at line: 421.

(vi) This OSL source code (" Color3 path_radiance(0, 0, 0)") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code (" Color3 path_radiance(0, 0, 0)") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0009 through line 0011.

121

(J)

Coffelt: 0012 blueD += shiftd;
0013 greenD += shiftd;
0014 redD += shiftd;

OSL: path_radiance += path_weight * bg

Basis: (i) Coffelt's source code ("blueD +=, greenD +=, redD += ") in the Complaint at paragraph 66, line 0012 through line 0014 is shifting the base color of the surface by the numerical quanity ("shiftd").

(ii) OSL source code ("path_radiance += path_weight * bg") is shifting the base color of the surface by the numerical quanity ("path_weight * bg").

(iii) OSL source code ("path_radiance") is type ("Color3") having 3 components, Red, Green, Blue. *See* file name: ("OpenShadingLanguage-master\src\testrender\testrender.cpp") at line: 421.

(iv) Sony Imageworks publication confirms OSL is based on direction of view and reflection vector See EXHIBIT 123.

(v) Comment in OSL file ("OpenShadingLanguage-master\OpenShadingLanguage-master\src\testrender\shading.cpp") at line 178 ("// reflect the view vector") confirms code is directed to the reflection vector and direction of view.

(vi) The location of this OSL source code ("path_radiance += path_weight * bg") is file name: ("OpenShadingLanguage-master\src\testrender\testrender.cpp") at line: 434.

(vii) This OSL source code ("path_radiance += path_weight * bg") is available to the public at: https://github.com/imageworks/OpenShadingLanguage

For these reasons, the OSL source code (" path_radiance += path_weight * bg") is equivalent to Coffelt's source code in the Complaint at paragraph 66, line 0012 through line 0014.

For all of the above reasons, and those in the Complaint, the above identified OSL source code in items (A) through (J) is an unauthorized derivative work based on Coffelt's copyrighted Gradient Work, U.S. Registration No. TXu002049564, Date of registration: On June 12, 2017, year created: 2013

For all of the above reasons, and those in the Complaint, the above identified OSL source code in items (A) through (J) is an unauthorized derivative work based on Coffelt's copyrighted Photorealistic Gradient Work, U.S. Application No.: 1-5376971191, filed: On June 12, 2017, year created: 2013

Coffelt's Gradient Work is a derivative work based on Coffelt's Gradient Work 2010.

Coffelt's Photorealistic Gradient Work is a derivative work based on Coffelt's Gradient Work 2010.

For all of the above reasons, and those in the Complaint, the above identified OSL source code in items (A) through (J) is an unauthorized derivative work based on Coffelt's copyrighted Gradient Work 2010, U.S. Application No.: 1-5121154211, filed: May 13, 2017, year created: 2010

EXHIBIT 122

Your account has been flagged.

Because of that, your profile is hidden from the public. If you believe this is a mistake, contact support to have your account status reviewed.

imageworks / **OpenShadingLanguage**

Advanced shading language for production GI renderers

#osl #shading-language #shaders #c-plus-plus #computer-graphics #computer-language #llvm

1,997 commits **14** branches **82** releases **29** contributors BSD-3-Clause

Branch: master ▾ | New pull request | Create new file | Upload files | Find file | Clone or download ▾

lgritz Trim verbose build output of clutter — Latest commit 61bb74e 13 days ago

.github	Minor formatting revision of PR and Issue templates	7 months ago
site	Fixup for SPI to put the libs in our usual place	2 months ago
src	Trim verbose build output of clutter	7 days ago
testsuite	ctransform full derivative support.	a month ago
.gitignore	testsuite overhaul -- all temp files end up in build, runtest.py call...	6 years ago
.travis.yml	Bump LLVM minimum from 3.5 to 3.9. (#806)	9 days ago
CHANGES.md	Bump LLVM minimum from 3.5 to 3.9. (#806)	9 days ago
CMakeLists.txt	Simplify the PugiXML logic. (#809)	14 days ago
CONTRIBUTING.md	Docs for PR, Issue, Contributing	7 months ago
INSTALL.md	Bump LLVM minimum from 3.5 to 3.9. (#806)	9 days ago
LICENSE	Docs	10 months ago
Makefile	Simplify the PugiXML logic. (#809)	14 days ago

521

README md CHANGES 24 days ago

README.md

Open Shading Language

Build status:

Table of contents

Introduction

Welcome to Open Shading Language!

021

Open Shading Language (OSL) is a small but rich language for programmable shading in advanced renderers and other applications, ideal for describing materials, lights, displacement, and pattern generation.

OSL was originally developed by Sony Pictures Imageworks for use in its in- house renderer used for feature film animation and visual effects, released as open source so it could be used by other visual effects and animation studios and rendering software vendors. Now it's the de facto standard shading language for VFX and animated features, used across the industry in many commercial and studio- proprietary renderers. Because of this, the work on OSL received an Academy Award for Technical Achievement in 2017.

OSL is robust and production-proven, and has been used in films as diverse as "The Amazing Spider-Man," "Hotel Transylvania," "Edge of Tomorrow", "Ant Man", "Finding Dory," and many more. OSL support is in most leading renderers used for high-end VFX and animation work. For a full list of films and products, see the filmography.

The OSL code is distributed under the "New BSD" license, and the documentation under the Creative Commons Attribution 3.0 Unported License. In short, you are free to use OSL in your own applications, whether they are free or commercial, open or proprietary, as well as to modify the OSL code and documentation as you desire, provided that you retain the original copyright notices as described in the license.

How OSL is different

OSL has syntax similar to C, as well as other shading languages. However, it is specifically designed for advanced rendering algorithms and has features such as radiance closures, BSDFs, and deferred ray tracing as first-class concepts.

OSL has several unique characteristics not found in other shading languages (certainly not all together). Here are some things you will find are different in OSL compared to other languages:

- Surface and volume shaders compute radiance closures, not final colors.

  OSL's surface and volume shaders compute an explicit symbolic description, called a "closure", of the way a surface or volume scatters light, in units of radiance. These radiance closures may be evaluated in particular directions, sampled to find important directions, or saved for later evaluation and re-evaluation. This new approach is ideal for a physically-based renderer that supports ray tracing and global illumination.

127

In contrast, other shading languages usually compute just a surface color as visible from a particular direction. These old shaders are "black boxes" that a renderer can do little with but execute to find this one piece of information (for example, there is no effective way to discover from them which directions are important to sample). Furthermore, the physical units of lights and surfaces are often underspecified, making it very difficult to ensure that shaders are behaving in a physically correct manner.

- Surface and volume shaders do not loop over lights or shoot rays.

  There are no "light loops" or explicitly traced illumination rays in OSL surface shaders. Instead, surface shaders compute a radiance closure describing how the surface scatters light, and a part of the renderer called an "integrator" evaluates the closures for a particular set of light sources and determines in which directions rays should be traced. Effects that would ordinarily require explicit ray tracing, such as reflection and refraction, are simply part of the radiance closure and look like any other BSDF.

  Advantages of this approach include that integration and sampling may be batched or re-ordered to increase ray coherence; a "ray budget" can be allocated to optimally sample the BSDF; the closures may be used by for bidirectional ray tracing or Metropolis light transport; and the closures may be rapidly re-evaluated with new lighting without having to re-run the shaders.

- Surface and light shaders are the same thing.

  OSL does not have a separate kind of shader for light sources. Lights are simply surfaces that are emissive, and all lights are area lights.

- Transparency is just another kind of illumination.

  You don't need to explicitly set transparency/opacity variables in the shader. Transparency is just another way for light to interact with a surface, and is included in the main radiance closure computed by a surface shader.

- Renderer outputs (AOV's) may be specified using "light path expressions."

  Sometimes it is desirable to output images containing individual lighting components such as specular, diffuse, reflection, individual lights, etc. In other languages, this is usually accomplished by adding a plethora of "output variables" to the shaders that collect these individual quantities.

OSL shaders need not be cluttered with any code or output variables to accomplish this. Instead, there is a regular-expression-based notation for describing which light paths should contribute to which outputs. This is all done on the renderer side (though supported by the OSL implementation). If you desire a new output, there is no need to modify the shaders at all; you only need to tell the renderer the new light path expression.

- Shaders are organized into networks.

  OSL shaders are not monolithic, but rather can be organized into networks of shaders (sometimes called a shader group, graph, or DAG), with named outputs of some nodes being connected to named inputs of other nodes within the network. These connections may be done dynamically at render time, and do not affect compilation of individual shader nodes. Furthermore, the individual nodes are evaluated lazily, only when their outputs are "pulled" from the later nodes that depend on them (shader writers may remain blissfully unaware of these details, and write shaders as if everything is evaluated normally).

- Arbitrary derivatives without grids or extra shading points.

  In OSL, you can take derivatives of any computed quantity in a shader, and use arbitrary quantities as texture coordinates and expect correct filtering. This does not require that shaded points be arranged in a rectangular grid, or have any particular connectivity, or that any "extra points" be shaded. This is because derivatives are not computed by finite differences with neighboring points, but rather by "automatic differentiation", computing partial differentials for the variables that lead to derivatives, without any intervention required by the shader writer.

- OSL optimizes aggressively at render time

  OSL uses the LLVM compiler framework to translate shader networks into machine code on the fly (just in time, or "JIT"), and in the process heavily optimizes shaders and networks with full knowledge of the shader parameters and other runtime values that could not have been known when the shaders were compiled from source code. As a result, we are seeing our OSL shading networks execute 25% faster than the equivalent shaders hand-crafted in C! (That's how our old shaders worked in our renderer.)

What OSL consists of

The OSL open source distribution consists of the following components:

- oslc, a standalone compiler that translates OSL source code into an assembly-like intermediate code (in the form of .oso files).
- liboslc, a library that implements the OSLCompiler class, which contains the guts of the shader compiler, in case anybody needs to embed it into other applications and does not desire for the compiler to be a separate executable.
- liboslquery, a library that implements the OSLQuery class, which allows applications to query information about compiled shaders, including a full list of its parameters, their types, and any metadata associated with them.
- oslinfo, a command-line program that uses liboslquery to print to the console all the relevant information about a shader and its parameters.
- liboslexec, a library that implements the ShadingSystem class, which allows compiled shaders to be executed within an application. Currently, it uses LLVM to JIT compile the shader bytecode to x86 instructions.
- testshade, a program that lets you execute a shader (or connected shader network) on a rectangular array of points, and save any of its outputs as images. This allows for verification of shaders (and the shading system) without needing to be integrated into a fully functional renderer, and is the basis for most of our testsuite verification. Along with testrender, testshade is a good example of how to call the OSL libraries.
- testrender, a tiny ray-tracing renderer that uses OSL for shading. Features are very minimal (only spheres are permitted at this time) and there has been no attention to performance, but it demonstrates how the OSL libraries may be integrated into a working renderer, what interfaces the renderer needs to supply, and how the BSDFs/radiance closures should be evaluated and integrated (including with multiple importance sampling).
- A few sample shaders.
- Documentation -- at this point consisting of the OSL language specification (useful for shader writers), but in the future will have detailed documentation about how to integrate the OSL libraries into renderers.

Where OSL has been used

130

This list only contains films or products whose OSL use is stated or can be inferred from public sources, or that we've been told is ok to list here. If an OSL-using project is missing and it's not a secret, just email the OSL project leader or submit a PR with edits to this file.

Renderers and other tools with OSL support (in approximate order of adding OSL support):

- Sony Pictures Imageworks: in-house "Arnold" renderer
- Blender/Cycles
- Chaos Group: V-Ray
- Pixar: PhotoRealistic RenderMan RIS
- Isotropix: Clarisse
- Autodesk Beast
- Appleseed
- Animal Logic: Glimpse renderer
- Image Engine: Gaffer (for expressions and deformers)
- DNA Research: 3Delight
- Ubisoft motion picture group's proprietary renderer
- Autodesk/SolidAngle: Arnold

Films using OSL (grouped by year of release date):

- **(2012)** Men in Black 3, The Amazing Spider-Man, Hotel Transylvania
- **(2013)** Oz the Great and Powerful, Smurfs 2, Cloudy With a Chance of Meatballs 2
- **(2014)** The Amazing Spider-Man 2, Blended, Edge of Tomorrow, 22 Jump Street, Guardians of the Galaxy, Fury, The Hunger Games: Mockingjay - Part 1, Exodus: Gods and Kings, The Interview
- **(2015)** American Sniper, Insurgent, Avengers Age of Ultron, Ant Man, Pixels, Mission Impossible: Rogue Nation, Hotel Transylvania 2, Bridge of Spies, James Bond: Spectre, The Hunger Games: Mockingjay - Part 2, Concussion
- **(2016)** Allegiant, Batman vs Superman: Dawn of Justice, The Huntsman, Angry Birds Movie, Alice Through the Looking Glass, Captain America: Civil War, Finding Dory, Piper, Independence Day: Resurgence, Ghostbusters, Star Trek Beyond, Suicide Squad, Kingsglaive: Final Fantasy XV, Storks, Miss Peregrine's Home for Peculiar Children, Assassin's Creed

131

- **(2017) / upcoming** Lego Batman, The Great Wall, A Cure for Wellness, Logan, Power Rangers, Life, Smurfs: The Lost Village, The Fate of the Furious, Alien Covenant, Guardians of the Galaxy 2, The Mummy, Wonder Woman, Cars 3, Baby Driver, Spider-Man: Homecoming, Dunkirk, The Emoji Movie, Detroit, Kingsman: The Golden Circle, Lega Ninjago Movie, Blade Runner 2049, Geostorm, ...

Building OSL

Please see the INSTALL.md file in the OSL distribution for instructions for building the OSL source code.

Contacts, Links, and References

OSL GitHub page

Read or subscribe to the OSL development mail list

Email the lead architect: lg AT imageworks DOT com

Most recent PDF of the OSL language specification

OSL home page at SPI

Sony Pictures Imageworks main open source page

If you want to contribute code back to the project, you'll need to sign a Contributor License Agreement.

Credits

The original designer and project leader of OSL is Larry Gritz. Other early developers of OSL are (in order of joining the project): Cliff Stein, Chris Kulla, Alejandro Conty, Jay Reynolds, Solomon Boulos, Adam Martinez, Brecht Van Lommel.

Additionally, many others have contributed features, bug fixes, and other changes: Steve Agland, Shane Ambler, Martijn Berger, Farchad Bidgolirad, Nicholas Bishop, Stefan Büttner, Matthaus G. Chajdas, Thomas Dinges, Henri Fousse, Syoyo Fujita, Derek Haase, Sven-Hendrik Haase, John Haddon, Daniel Heckenberg, Ronan Keryell, Elvic Liang, Max Liani, Bastien Montagne, Erich Ocean, Mikko Ohtamaa, Alex Schworer, Sergey Sharybin, Stephan Steinbach, Esteban Tovagliari, Alexander von Knorring, Roman Zulak. (Listed alphabetically; if we've left anybody out, please let us know.)

We cannot possibly express sufficient gratitude to the managers at Sony Pictures Imageworks who allowed this project to proceed, supported it wholeheartedly, and permitted us to release the source, especially Rob Bredow, Brian Keeney, Barbara Ford, Rene Limberger, and Erik Strauss.

Huge thanks also go to the crack shading team at SPI, and the brave lookdev TDs and CG supes willing to use OSL on their shows. They served as our guinea pigs, inspiration, testers, and a fantastic source of feedback. And of course, the many engineers, TDs, and artists elsewhere who incorporated OSL into their products and pipelines, especially the early risk-takers at Chaos Group, Double Negative, Pixar, DNA, Isotropix, and Animal Logic. Thank you, and we hope we've been responsive to your needs.

OSL was not developed in isolation. We owe a debt to the individuals and studios who patiently read early drafts of the language specification and gave us very helpful feedback and additional ideas, as well as to the continuing contributions and feedback of its current developers and users at other VFX and animation studios.

The OSL implementation depends upon several other open source packages, all with compatible licenses:

- OpenImageIO (c) Larry Gritz, et al
- Boost - various authors
- IlmBase (c) Industrial Light & Magic
- LLVM Compiler Infrastructure

OSL's documentation incorporates parts of Markdeep (c) 2015-2016, Morgan McGuire, and highlight.js (c) 2006, Ivan Sagalaev, both distributed under BSD licenses.

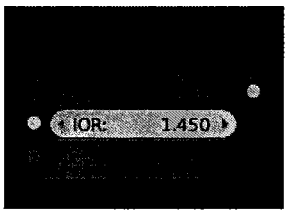
EXHIBIT 123

Date Downloaded: May 20, 2017
Computer File Location: Coffelt's Laptop

Service Tag: GMBTY32

Fresnel Node

The *Fresnel* or *Dielectric Fresnel* node computes how much light is reflected off a layer, where the rest will be refracted through the layer. The resulting weight can be used for layering shaders with the *Mix Shader* node. It is dependent on the angle between the surface normal and the viewing direction.



Fresnel Node.

The most common use is to mix between two BSDFs using it as a blending factor in a mix shader node. For a simple glass material you would mix between a glossy refraction and glossy reflection. At grazing angles more light will be reflected than refracted as happens in reality.

For a two-layered material with a diffuse base and a glossy coating, you can use the same setup, mixing between a diffuse and glossy BSDF. By using the Fresnel as the blending factor you are specifying that any light which is refracted through the glossy coating layer would hit the diffuse base and be reflected off that.

Inputs

IOR
Index of refraction (IOR) of the material being entered.

Normal
Input meant for plugging in bump or normal maps which will affect the output.

Properties

This node has no properties.

Outputs

Factor
Fresnel weight, indicating the probability with which light will reflect off the layer rather than passing through.

Source: https://docs.blender.org/manual/en/dev/render/cycles/nodes/types/input/fresnel.html

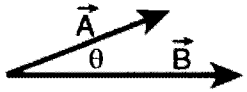
EXHIBIT 124

Scalar Product of Vectors

The scalar product and the vector product are the two ways of multiplying vectors which see the most application in physics and astronomy. The scalar product of two vectors can be constructed by taking the component of one vector in the direction of the other and multiplying it times the magnitude of the other vector. This can be expressed in the form:

$$\vec{A}\cdot\vec{B} = A\,B\cos\theta$$

Calculation



$\vec{A}$ denotes vector

A denotes the magnitude of the vector.

If the vectors are expressed in terms of unit vectors i, j, and k along the x, y, and z directions, the scalar product can also be expressed in the form:

$$\vec{A}\cdot\vec{B} = A_x B_x + A_y B_y + A_z B_z \quad \text{where} \quad \vec{A} = A_x\vec{i} + A_y\vec{j} + A_z\vec{k}, \quad \vec{B} = B_x\vec{i} + B_y\vec{j} + B_z\vec{k}$$

Applications

The scalar product is also called the "inner product" or the "dot product" in some mathematics texts.

Matrix approach to scalar product

Index

Vector concepts

HyperPhysics***** Mechanics R Nave

Go Back

Scalar Product Calculation

Index

Vector concepts

You may enter values in any of the boxes below. Then click on the symbol for either the scalar product or the angle. The vectors A and B cannot be unambiguously calculated from

EXHIBIT 125

function template

std::max

<algorithm>

C++98 | C++11 | C++14

```
default (1)          template <class T> constexpr const T& max (const T& a, const T& b);
custom (2)           template <class T, class Compare>
                       constexpr const T& max (const T& a, const T& b, Compare comp);
initializer list (3) template <class T> constexpr T max (initializer_list<T> il);
                     template <class T, class Compare>
                       constexpr T max (initializer_list<T> il, Compare comp);
```

Return the largest

Returns the largest of *a* and *b*. If both are equivalent, *a* is returned.

The versions for *initializer lists (3)* return the largest of all the elements in the list. Returning the first of them if these are more than one.

The function uses `operator<` (or *comp*, if provided) to compare the values.

The behavior of this function template (C++98) is equivalent to:

```
1 template <class T> const T& max (const T& a, const T& b) {
2   return (a<b)?b:a;     // or: return comp(a,b)?b:a; for version (2)
3 }
```

Parameters

`a, b`
: Values to compare.

`comp`
: Binary function that accepts two values of type `T` as arguments, and returns a value convertible to `bool`. The value returned indicates whether the element passed as first argument is considered less than the second.

EXHIBIT 126

function template

std::min

<algorithm>

C++98 | C++11 | C++14

```
default (1)          template <class T> constexpr const T& min (const T& a, const T& b);
custom (2)           template <class T, class Compare>
                       constexpr const T& min (const T& a, const T& b, Compare comp);
initializer list (3) template <class T> constexpr T min (initializer_list<T> il);
                     template <class T, class Compare>
                       constexpr T min (initializer_list<T> il, Compare comp);
```

Return the smallest

Returns the smallest of *a* and *b*. If both are equivalent, *a* is returned.

The versions for *initializer lists (3)* return the smallest of all the elements in the list. Returning the first of them if these are more than one.

The function uses `operator<` (or *comp*, if provided) to compare the values.

The behavior of this function template (C++98) is equivalent to:

```
1 template <class T> const T& min (const T& a, const T& b) {
2   return !(b<a)?a:b;     // or: return !comp(b,a)?a:b; for version (2)
3 }
```

Parameters

`a, b`
: Values to compare.

`comp`
: Binary function that accepts two values of type `T` as arguments, and returns a value convertible to `bool`. The value returned indicates whether the element passed as first argument is considered less than the second.

141

EXHIBIT 127

Interview: Larry Gritz - Lead Developer of OSL (/en/interviews/531-interview-larry-gritz-lead-developer-of-osl.html)

Written by Gottfried Hofmann

Print (/en/interviews/531-interview-larry-gritz-lead-developer-of-osl.html?tmpl=component&print=1&layout=default)

Email (/en/component/mailto/?tmpl=component&template=sj_plus&link=91b2aa03e5cc21d0c3d0668f6f9df39a2e205f20)

Published: 10 October 2013

Larry Gritz, lead developer of the Open Shading Language at SPI

BlenderDiplom interviews Larry Gritz, lead developer of the Open Shading Language (OSL) at Sony Pictures Imageworks. OSL has been sticking around in Cycles for a while and is supported by the latest version of V-Ray which is currently in Beta. This interview was conducted in May 2013 and was first released in German language in the magazine Digital Production (http://www.digitalproduction.com/).

BD: What changes did OSL introduce to the shading workflow at Sony Pictures Imageworks?

LG: Prior to OSL, SPI's shaders were compiled C plugins for Arnold. Our shader writers were spending more and more time dealing with low-level details -- C is rather clunky to use as a shading language, and very error-prone -- and dealing with headaches of interfacing with the renderer internals. Furthermore, it was increasingly difficult to achieve the kind of physically-based lights and materials that we wanted, and it was clear that we needed a different set of abstractions than had been used in prior shading systems.

OSL really changes three things for us at once: First, because the syntax and semantics of the language are designed with shading in mind, it is simply a more convenient notation for describing shading (especially compared to raw C). Second, it presents a really clean API that doesn't directly expose ugly or complex renderer internals. These two things put together mean that nearly all of the lines of code of a shader -- and the time and attention of the shader writer -- are about describing the material, and a great amount of work that was shuffling data around and dealing with renderer internals is eliminated. Finally, there is a big conceptual shift, from computing the appearance from one direction, which would have to include all the ugly details of sampling and integration in the shader itself, to computing view-independent material closures (which leave the sampling and integration up to the renderer, which is much better equipped to do it well). That has been a big enabler of our efforts to have more physically-based light and material models and get more pleasing and accurate images "out of the box."

BD: What were the motivations behind open sourcing OSL?

LG: Proprietary tools sometimes have a competitive advantage, but they also have a real cost: any tool or format that is truly unique to your facility is one where you can never hire somebody who already is an expert, you will never find external documentation, you will not have your 3rd party tools support it without a lot of additional plug-in development that you have to take on yourself.

So we were betting that any advantage of keeping it to ourselves would be small compared to the really big benefits of having an OSL ecosystem that extended beyond just SPI. We envisioned a day when 3rd party tools we use would support it, when we could hire TDs who were already familiar with this method of shading. It's even an advantage when hiring people who don't already know OSL, because it's a fact of life that artists move from facility to facility, and it's more appealing for a TD to learn skills that they feel will be valuable even after leaving Sony, than learning details about tools they'll never see again.

Also, it's very professionally satisfying for the software developers, who often don't get film credits and can't easily point to particular shots they worked on, to be working on things that are externally visible and get recognition from their peers at other VFX facilities.
Knowing that people will actually see your work makes you put more effort into making solid code you can be proud of. So we really believe that by developing OSL as an open source project, the implementation is better.

BD: Did you get many contributions from outside yet?

LG: Yes! Even before OSL was implemented, we circulated early drafts to some other interested studios and got lots of great early feedback even from other major well-known studios.

In terms of actual code contributions, the main feature authors are at SPI, but we have had many contributions from elsewhere. Especially now that OSL has been incorporated into Blender/Cycles, V-Ray, and Autodesk Beast, we regularly get contributions from the authors of those packages that include minor bug fixes, optimizations, and changes to the code that allow smoother porting and compilation for platforms we don't tend to use ourselves (particularly aid to getting it running nicely on Windows).

BD: OSL was open sourced in 2010, now we have 2013 and still Blender Cycles is the only render engine on the open market that supports OSL. Are the developers of render engines hesitant to adopt it and if so, what do you think are the reasons?

LG: That's not quite correct; it's also incorporated into Autodesk Beast and being incorporated into V-Ray, both of which have substantial user bases.

Also, although OSL has been open sourced since 2010 (in the sense that we did much of the initial development out in the open, even before it was completed), it was only mid-2012 that SPI completed the first "all-OSL" films, which was an important milestone that lots of people were waiting for before taking a gamble on it. So it's really only been one year of what you'd consider a 1.0, fully production-hardened state, and 3 major engines have incorporated OSL to date that we are aware of.

It's hard to switch an existing render engine to OSL for three reasons:
(1) the shader execution engine (and almost certainly the texture system, along with it) is a very large portion of a renderer to replace with an external package that you don't necessarily have full control over.
(2) "upgrading" an existing renderer to OSL may also involve a substantial overhaul of how lights and sampling work even in the non-OSL parts of the renderer; (3) products with large existing customer bases have a lot of inertia behind their user education, expertise, and shader libraries, so it's not a small task to pivot to a different way of specifying shading.

BD: The syntax is really similar to RenderMan Shading Language, did you choose that so users can port their existing shaders more easily?

LG: RenderMan Shading Language did a lot of things right, we wanted to build on that where it worked, but did not hesitate to deviate where we could improve. We didn't really worry about porting existing shaders, but we certainly wanted to build on what shader programmers already know of existing programming languages. RSL in turn looks very similar to C, as do most of the widely used shading languages (GLSL, Cg, Mantra, etc.), for similar reasons. There's no advantage to starting out with a language syntax that is wildly dissimilar to anything that shader writers are used to.

(https://www.f	(https://twitt	(https://plus.g	(https://www.s	(https://www.l	(https://digg	(http://reddit.c	(http://pintere	(mailto:?
app_id=8789	text=Intervie	url=http%3A%	url=http%3A%	mini=true&url	phase=2&u	url=http%3A%	url=http%3A%	subject=I
interview-	%20Lead%	interview-	interview-	interview-	interview-	interview-	interview-	thought

Comments (6) Add New

Loading comments

Leave a comme

Posting as Anonymous

Notify me of followup comments via e-mail

Enter your name

Enter your email

Displayed next to your comments.

Not displayed publicly. Gravatar (http://gravatar.com) enabled

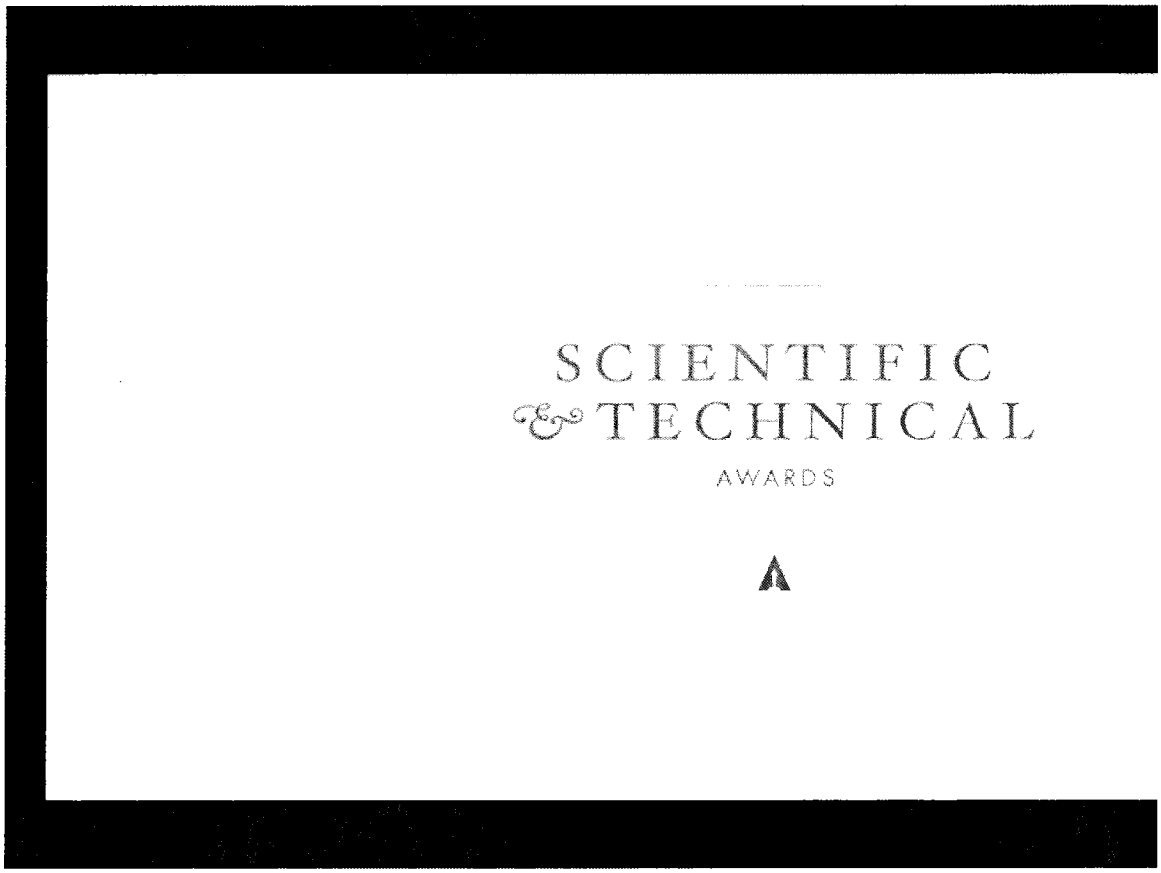
Type the code that you see in the image

941

EXHIBIT 128

Date Downloaded: August 9, 2017
Computer File Location: Coffelt's Laptop
Service Tag: GMBTY32

18 SCIENTIFIC AND TECHNICAL ACHIEVEMENTS TO BE HONORED WITH ACADEMY AWARD



Posted:
Wednesday, January 4, 2017 - 12:30

The Academy of Motion Picture Arts and Sciences announced today that 18 scientific and technical achiever award recipients, as well as five organizations, will be honored at its annual Scientific and Technical Awards F 11, 2017 at the Beverly Wilshire in Beverly Hills.

"This year we are particularly pleased to be able to honor not only a wide range of new technologies, but also cameras that helped facilitate the widespread conversion to electronic image capture for motion picture pro Award® recipient and chair of the Scientific and Technical Awards Committee. "With their outstanding, inn engineers and inventors have significantly expanded filmmakers' creative choices for moving image storytelli

Source:

Unlike other Academy Awards to be presented this year, achievements receiving Scientific and Technical Aw and introduced during 2016. Rather, the achievements must demonstrate a proven record of contributing sig making motion pictures.

The Academy Awards for scientific and technical achievements are:

TECHNICAL ACHIEVEMENT AWARDS (ACADEMY CERTIFICATES)

To Thomson Grass Valley for the design and engineering of the pioneering Viper FilmStream digita

The Viper camera enabled frame-based logarithmic encoding, which provided uncompressed came into existing digital intermediate workflows.

To Larry Gritz for the design, implementation and dissemination of Open Shading Language (OSL)

OSL is a highly optimized runtime architecture and language for programmable shading and texturi industry standard. It enables artists at all levels of technical proficiency to create physically plausibl rendering.

To Carl Ludwig, Eugene Troubetzkoy and Maurice van Swaaij for the pioneering development of th Studios.

CGI Studio's groundbreaking ray-tracing and adaptive sampling techniques, coupled with streamlin the feasibility of ray-traced rendering for feature film production.

To Brian Whited for the design and development of the Meander drawing system at Walt Disney A

Meander's innovative curve-rendering method faithfully captures the artist's intent, resulting in a si communication throughout the production pipeline.

To Mark Rappaport for the concept, design and development, to Scott Oshita for the motion analy for the development of the faux-hair finish techniques, and to Todd Minobe for the character articu of the Creature Effects Animatronic Horse Puppet.

The Animatronic Horse Puppet provides increased actor safety, close integration with live action, a filmmakers.

To Glenn Sanders and Howard Stark for the design and engineering of the Zaxcom Digital Wireless

The Zaxcom system has advanced the state of wireless microphone technology by creating a fully di feature set, which includes local recording capability within the belt pack and a wireless control sch control and time-code distribution.

To David Thomas, Lawrence E. Fisher and David Bundy for the design, development and engineeri Hybrid Wireless Microphone System.

The Lectrosonics system has advanced the state of wireless microphone technology by means of an algorithm to realize full fidelity audio transmission over a conventional analog FM radio link, by red increasing power efficiency.

To Parag Havaldar for the development of expression-based facial performance-capture technology

EXHIBIT 129

Date Downloaded: July 26, 2017
Computer File Location: Coffelt's Laptop

Service Tag: GMBTY32

LATEST NEWS

13 FEB 2017 4:25 am

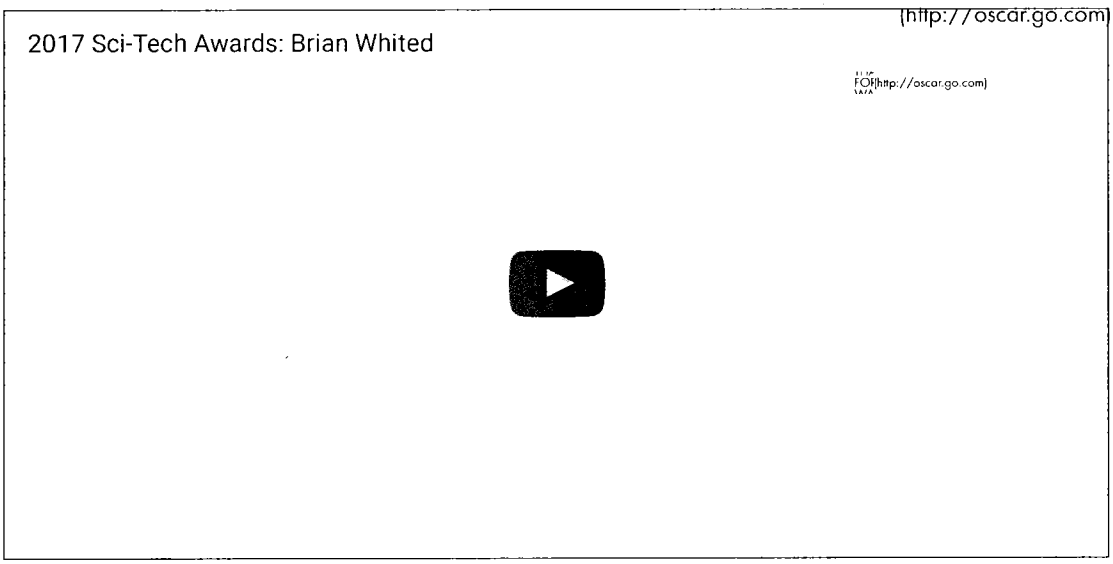
THE ACADEMY'S SCI-TECH AWARDS 2017 WINNERS: SEE THE FULL LIST

BY. CHARLES SCHUMLEY

On Saturday, February 11 at the Beverly Wilshire Hotel, John Cho and Leslie Mann hosted the annual Scientific and Technical Awards Presentation (/news/oscar-news/john-cho-and-leslie-mann-will-host-the-2017-sci-tech-awards) honoring 10 scientific and technical achievements represented by 35 individual award recipients.

You can see more highlights (https://www.youtube.com/watch?v=3RGThG-X-gU&list=PLJ8RjvesnvDML8hl5NWWflitE0DefRe5L)and the full list of honorees below.

Source:



The Academy Awards for scientific and technical achievements are:

TECHNICAL ACHIEVEMENT AWARDS (ACADEMY CERTIFICATES)

To **Thomson Grass Valley** for the design and engineering of the pioneering Viper FilmStream digital camera system.

The Viper camera enabled frame-based logarithmic encoding, which provided uncompressed camera output suitable for importing into existing digital intermediate workflows.

To **Larry Gritz** for the design, implementation and dissemination of Open Shading Language (OSL).

OSL is a highly optimized runtime architecture and language for programmable shading and texturing that has become a de facto industry standard. It enables artists at all levels of technical proficiency to create physically plausible materials for efficient production rendering.

To **Carl Ludwig**, **Eugene Troubetzkoy** and **Maurice van Swaaij** for the pioneering development of the CGI Studio renderer at Blue Sky Studios.

CGI Studio's groundbreaking ray-tracing and adaptive sampling techniques, coupled with streamlined artist controls, demonstrated the feasibility of ray-traced rendering for feature film production.

To **Brian Whited** for the design and development of the Meander drawing system at Walt Disney Animation Studios.

Meander's innovative curve-rendering method faithfully captures the artist's intent, resulting in a significant improvement in creative communication throughout the production pipeline.

To **Mark Rappaport** for the concept, design and development, to **Scott Oshita** for the motion analysis and CAD design, to **Jeff Cruts** for the development of the faux-hair finish techniques, and to **Todd Minobe** for the character articulation and drive-train mechanisms, of the Creature Effects Animatronic Horse Puppet.

The Animatronic Horse Puppet provides increased actor safety, close integration with live action, and improved realism for filmmakers.

To **Glenn Sanders** and **Howard Stark** for the design and engineering of the Zaxcom Digital Wireless Microphone System.

The Zaxcom system has advanced the state of wireless microphone technology by creating a fully digital modulation system with a rich feature set, which includes local recording capability within the belt pack and a wireless control scheme providing real-time transmitter control and time-code distribution.

To **David Thomas**, **Lawrence E. Fisher** and **David Bundy** for the design, development and engineering of the Lectrosonics Digital Hybrid Wireless Microphone System.

The Lectrosonics system has advanced the state of wireless microphone technology by means of an innovative digital predictive algorithm to realize full fidelity audio transmission over a conventional analog FM radio link, by reducing transmitter size, and by increasing power efficiency.

To **Parag Havaldar** for the development of expression-based facial performance-capture technology at Sony Pictures Imageworks.

This pioneering system enabled large-scale use of animation rig-based facial performance-capture for motion pictures, combining solutions for tracking, stabilization, solving and animator-controllable curve editing.

To **Nicholas Apostoloff** and **Geoff Wedig** for the design and development of animation rig-based facial performance-capture systems at ImageMovers Digital and Digital Domain.

These systems evolved through independent, then combined, efforts at two different studios, resulting in an artist-controllable, editable, scalable solution for the high-fidelity transfer of facial performances to convincing digital characters.

To **Kiran Bhat**, **Michael Koperwas**, **Brian Cantwell** and **Paige Warner** for the design and development of the ILM facial performance-capture solving system.

This system enables high-fidelity facial performance transfer from actors to digital characters in large-scale productions while retaining full artistic control, and integrates stable rig-based solving and the resolution of secondary detail in a controllable pipeline.

SCIENTIFIC AND ENGINEERING AWARDS (ACADEMY PLAQUES)

To **ARRI** for the pioneering design and engineering of the Super 35 format Alexa digital camera system.

With an intuitive design and appealing image reproduction, achieved through close collaboration with filmmakers, ARRI's Alexa cameras were among the first digital cameras widely adopted by cinematographers.

To **RED Digital Cinema** for the pioneering design and evolution of the RED Epic digital cinema cameras with upgradeable full-frame image sensors.

RED's revolutionary design and innovative manufacturing process have helped facilitate the wide adoption of digital image capture in the motion picture industry.

To **Sony** for the development of the F65 CineAlta camera with its pioneering high-resolution imaging sensor, excellent dynamic range, and full 4K output.

Sony's unique photosite orientation and true RAW recording deliver exceptional image quality.

To **Panavision** and **Sony** for the conception and development of the groundbreaking Genesis digital motion picture camera.

Using a familiar form factor and accessories, the design features of the Genesis allowed it to become one of the first digital cameras to be adopted by cinematographers.

To **Marcos Fajardo** for the creative vision and original implementation of the Arnold Renderer, and to **Christopher Kulla**, **Alan King**, **Thiago Ize** and **Clifford Stein** for their highly optimized geometry engine and novel ray-tracing algorithms which unify the rendering of curves, surfaces, volumetrics and subsurface scattering as developed at Sony Pictures Imageworks and Solid Angle SL.

EXHIBIT 130

Date Downloaded: August 16, 2017
Computer File Location: Coffelt's Laptop
Service Tag: GMBTY32

BusinessWire
A Berkshire Hathaway Company

Autodesk

Autodesk Reveals New Gameware Advancements at GDC 2013

Game Development Solutions Boost UI Design, Artificial Intelligence, Lighting and Animation Tools

March 25, 2013 12:00 PM Eastern Daylight Time

SAN FRANCISCO--(BUSINESS WIRE)--**Game Developers Conference 2013 —** Autodesk, Inc. (NASDAQ:ADSK) unveiled new versions of its widely adopted game development middleware from the Autodesk Gameware product line that allow developers to create more compelling gaming experiences.

Updates for Autodesk Scaleform, Autodesk Navigation, Autodesk Beast and Autodesk HumanIK bring game developers enhanced compatibility for multiple platforms and game engines, increased functionality and powerful new tools. Autodesk Gameware is optimized for the current generation and certain next generation consoles.

"We are committed to helping developers of all sizes realize their creative visions. Our Gameware products allow artists to work near seamlessly across multiple platforms and engines, while removing traditional bottlenecks and accelerating production," said Chris Bradshaw, senior vice president, Media and Entertainment at Autodesk. "Our 2014 releases deliver advancements for the development community as they create new titles for both the current generation and next generation of hardware. We look forward to seeing what they come up with next."

Autodesk Scaleform 4.3 Enables Powerful UI Features Across Devices

Autodesk Scaleform 4.3 harnesses the power of the Adobe Flash toolset to help developers create immersive user interface (UI) environments while streamlining workflows and accelerating the development cycle. Used on projects ranging from blockbuster PC and console titles to social, casual and mobile games, it provides an artistic, design driven workflow to create hardware-accelerated 3D game menus, heads-up displays, animated textures, in-game videos and mini-games. With a versatile and proven toolset, Autodesk Scaleform is compatible across certain mobile platforms and is easily integrated with certain third party or in-house game engines.

New features of Scaleform 4.3 are support for additional ActionScript 3 application programming interface (API) classes to help increase platform compatibility and native extensions for Adobe AIR, which enables developers to leverage an array of third party and community-developed extensions directly in Scaleform. The latest version also allows UI designers to leverage a wider range of color and shape transformations with improved Adobe Flash blending and has files demonstrating touch and gesture UI interaction. Scaleform 4.3 is compatible with the Unity 4 engine and also supports Microsoft Windows 8. Additional improvements are enhancements to mobile device compatibility to help facilitate the deployment of powerful UI features to a wider array of devices.

Autodesk Gameware Navigation 2014 Helps Improve Artificial Intelligence

With Autodesk Gameware Navigation 2014 middleware, developers can leverage full source code access, an accessible API and new remote visual debugging tools to create complex, ambitious artificial intelligence (AI) that allows for more gameplay. The successor to Autodesk Kynapse, Navigation has automatic NavMesh generation, pathfinding and path following in complex game environments. Character and obstacle avoidance, dynamic NavMesh, swappable sectors and Unreal Engine 3 are supported out-of-the-box with Gameware Navigation, greatly improving the speed and quality of AI iteration. The Gameware Navigation API is compatible with certain major gaming platforms.

Autodesk Beast 2014 Simulates Near Realistic In-Game Lighting

Autodesk Beast 2014 middleware enables game creators to simulate natural, physically near accurate lighting effects. Using Beast, artists can design and test lighting effects in game environments faster and more efficiently – helping result in more compelling game experiences. It can be integrated into level editors and allows artists to manipulate level lighting interactively by showing a highly accurate final render preview in almost real-time. The newest version of Beast adds physics-based rendering for more natural lighting, has Open Shading Language support to help increase flexibility in creating material properties and enables more near accurate preview renders. The Autodesk Beast API is designed to make integration with many game engines simple and intuitive.

Autodesk HumanIK 2014 Augments Character Animation

Autodesk HumanIK 2014 animation middleware helps reduce the need for developers to produce large libraries of character animations, saving more time for the creative. With full-body inverse kinematics and real-time retargeting technology, the latest version of HumanIK enables characters to interact with and procedurally adapt to game environments at run-time. Optimized to help facilitate the simultaneous animation of large groups of characters, HumanIK 2014 also adds improved support for mobile platforms, has easier control for squash and stretch deformations of the neck and spine and enhances quadruped animation support. As a C++ library, HumanIK supports many major gaming platforms and helps artists bring games to life with more near realistic and immersive character animation experiences.

Availability

Scaleform 4.3, Gameware Navigation 2014, Beast 2014 and HumanIK 2014 are expected to be available in spring 2013. To learn more about Autodesk's game development tools or access evaluation versions of the products, please visit: http://gameware.autodesk.com

About Autodesk

Autodesk, Inc., is a leader in 3D design, engineering and entertainment software. Customers across the manufacturing, architecture, building, construction, and media and entertainment industries -- including the last 18 Academy Award winners for Best Visual Effects -- use Autodesk software to design, visualize and simulate their ideas. Since its introduction of AutoCAD software in 1982, Autodesk continues to develop the broadest portfolio of state-of-the-art software for global markets. For additional information about Autodesk, visit www.autodesk.com.

Contacts

Autodesk, Inc.
Amelise Javier-Lane, 415-547-3562
amelise.javier.lane@autodesk.com
or
Karen Raz, 310-450-1482
karen@razpr.com

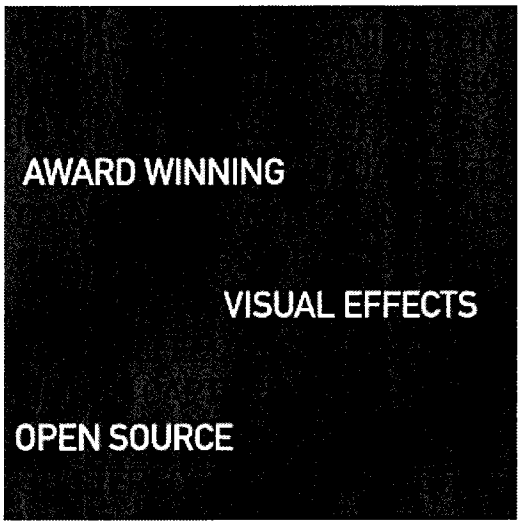
EXHIBIT 131

Date Downloaded: August 16, 2017
Computer File Location: Coffelt's Laptop

Service Tag: GMBTY32

Imageworks Open Source

- home
- contact us
- all projects
- imageworks.com



Community: One of the promises of Open Source. We're seeing the positive effects as a community of visual effects and animation professionals come together to solve problems more effectively today than ever before. This idea of giving stuff away is catching on and our industry is benefiting.

Our projects are seeing great adoption. Alembic, one of our most ambitious collaborations to date, is supported by most major 3d applications. OpenColorIO has also been widely adopted and is helping to simplify color pipelines in tools across our industry. Open Shading Language can be found in V-Ray, Autodesk Beast, Blender Cycles and other products coming soon. We're excited.

Please take a moment to check out these and our other open source offerings. They are provided with familiar, non-restrictive open source licenses and are already in use in studios around the world. These tools have already helped Sony Pictures Imageworks put films on the screen with greater ease, and we hope they can do the same for you.

- OSL

OSL

Open Shading Language

- Alembic

Alembic

Open Interchange Format

SOURCE

EXHIBIT 132

Download Date: August 13, 2017
Computer File Location: Coffelt's Laptop
Service Tag: GMBTY32

What's New in Beast 2013.2.x

May 30 2017

| In-Product View (http://help.autodesk.com/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-08210E19-206D-4643-96EE-24DFDEE68845.htm) (http://help.autodesk.com/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-08210E19-206D-4643-96EE-24DFDEE68845.htm)

| AUTODESK. Help

Applies to Beast 2015

What's New in Beast 2013.2.x

This page lists the new features and bug fixes introduced in the 2013.2 release.

Physically based rendering

This release introduces a totally revamped version of the Beast renderer, based around the concept of physical rendering. Some of the design goals for this new version were:

- The renderer should be easy to use, with minimal setup time for the artist.
- The live preview should match the final baking result.
- The old fixed function shading pipeline should be replaced with Open Shading Language (OSL).
- Performance should scale (almost) linearly with the number of computers you assign to your farm.

This is a major change that provides many new features, outlined below.

Physical scenes (https://www.autodesk.com/)

To accommodate all the new functionality offered by the physical renderer while preserving the old functionality, this release introduces a new type of scene called a *physical scene*. This is done to ensure that you do not use the old fixed-function materials and certain old render passes with the new physical renderer.

Rendering jobs that run on physical scenes currently support the following render passes:

- Full-shading
- Illumination
- IlluminationSH
- RNM

If your project requires the physical rendering system to support for any other passes, please contact Autodesk Support. See Support (https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-D70BFD6A-2A69-4C5D-9885-D6B1F176E60A-htm.html).

Physical materials and OSL

This release completely changes the way you set up materials in your scenes. Instead of calling API functions to set up a fixed-function shader built in to Beast, you now provide an OSL shader for each material. The shader is invoked by Beast when necessary to determine the effect of light on the surface. You can use the functions provided by the Beast API to set up input parameters for the shaders for each different material you create. For details, see Creating Physical Materials (https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-B812FA2F-A188-4D9A-A5A8-ACD7A771AA89-htm.html).

Live materials and material parameters

The new physical shading system allows you to change your materials and set new parameter values for your shaders during the course of a live session. This is a major improvement over the old materials system.

eRsnT and final render convergence

When using the physically based rendering system, the same renderer is used for both eRnsT and final render. Therefore, there should be no visible artifacts introduced in the final bake step that are not visible in the preview. (This excludes artifacts that are a part of a specific pass. For example, the RNM pass might introduce anomalies that come from applying the light information onto the final format of the pass).

In addition, there are no extra settings to tweak for the final render, which used to be done in the XML file you set for the job. For example, the primary cache, secondary cache, light sample settings, etc. The properties used to configure eRnsT and final baking are the same, except for quality settings.

Note, however, that due to the use of OSL shaders, this release only supports the "illumination only" mode when running eRnsT in live bake mode.

Post-filtering

One of the major changes in this release is the removal of all cache-based GI calculations when rendering physical scenes. This makes renders much easier to set up, without the need to spend time tweaking the behavior of the different render caches through the XML configuration.

Removing the caches also allows Beast to scale better in large render farms, since most of the sync points during a render job have been eliminated.

However, since we apply a path tracer approach, the raw result will likely contain more noise than in previous versions. To counter this noise, this release also introduces a new filter that can be applied for a physical job. This filter uses more data than a regular image filter, which means that you can get fairly good results even from quite noisy input data. The filter also filters between different shapes and bake types, which means that it can smooth out both UV-seams and seams between objects.

See Creating Render Targets (https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-C5DEE2AC-1BFE-4C80-8370-03BBFD925C4D-htm.html).

Maya plug-in

The Maya plug-in has been re-engineered to always produce a physical scene. As a convenience, the plug-in transparently converts Maya materials to an OSL representation. However, for best results, and the ability to tweak materials and material settings live, you are encouraged to set up your scene with custom shaders using the new BeastOSL node. The Beast Settings node has also been updated to reflect the simplified workflow. See the Maya plug-in (https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-1A6B5739-AA7F-4155-966A-54C81585442F-htm.html) section for details.

Improved frame rendering

This release introduces new functions for camera rendering:

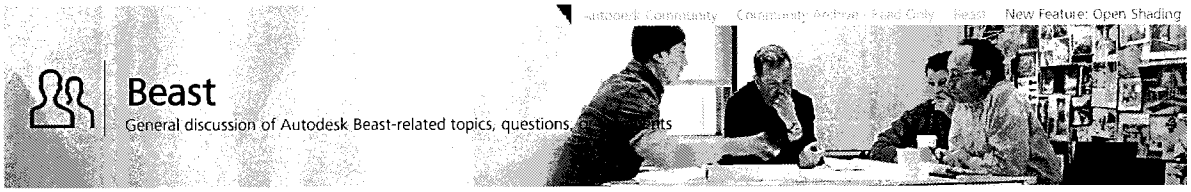
- ILBSetCameraDoF(), which allows you to set up the depth of field for a camera target.
- ILBSetCameraMotionTransforms(), which allows you to set up motion blurs for a camera target.

These new functions, combined with OSL support, allow you to output high quality camera renders from Beast.

Backward-compatibility

EXHIBIT 133

Date Downloaded: August 11, 2017
Computer File Location: Coffelt's Laptop
Service Tag: GMBTY32



nicolasleduc

602 Posts
42 Kudos
6 Solutions

Post 1 of 1

Report

New Feature: Open Shading Language Support

436 Views, 0 Replies
03-26-2013 06:07 AM

Options

This thread is to discuss the new Open Shading Language Support in Beast 2014. See a description of the feature here: Open Shading Language Support

SOURCE

EXHIBIT 134

Creating Physical Materials

Materials define the way the surfaces of your meshes react to light.

You define materials within a scene. When you create a mesh, you give each of its triangles a material by name. Then, when you create an instance of a mesh within a scene, each of its triangles uses the material in that scene whose name matches the name of the material assigned to it when the mesh was created. See Creating Meshes. You can also override the materials used in each instance of a mesh. See Creating Mesh Instances.

Beast uses a physical rendering system based on shaders expressed in the Open Shading Language (OSL). To customize the way each material interacts with light, you assign it a shader. Depending on the shader you use, you may also be able to set various input parameters (colors, textures, numeric values, etc.) for the material, which are passed to the shader as input values.

NOTE: The sections on this page detail the recommended way to set up materials in Beast, using the physically based rendering system introduced in release 2013.2. For instructions on using the older, fixed-shader materials system, see Creating Classic Materials.

What is the Open Shading Language?

Open Shading Language is an open-source shading system developed by Sony Pictures Imageworks for use in major feature films.

For more information, including source code, and a complete language specification that also outlines the library of functions you can use in your shaders, see: https://github.com/imageworks/OpenShadingLanguage (https://github.com/imageworks/OpenShadingLanguage)

OSL in Beast

Since OSL was designed for VFX film production, some aspects of it are not relevant for use when baking textures for games. In particular, Beast does not support:

- Displacement Shaders
- Volumetric Shaders
- BSSDRF shaders (sub surface scattering)
- Networks of OSL shaders

If you find that you need support for these or any other unsupported aspects of OSL, please contact Autodesk Support. See Support.

Writing an OSL shader file

In order to set up a material, you need to provide a file that contains an OSL shader. A shader can be as simple or as complex as you need in order to express the way your material surface interacts with light to produce its final shading. You can use the sample shader files provided in the *shaders* sub-directory of the Beast SDK package, or write your own from scratch.

For example, the following code shows a simple shader that multiplies the diffuse *closure* in OSL with a texture file that is passed as an input parameter.

```
Surface                                                  //< All Beast shaders use the Surface type.
diffuseTexture                                           //< A name for this shader.
( string diffuseFile = ""                                //< An input parameter.
     [[  string description = "Diffuse texture file" ]] ) //< Anything inside [[ ]] is optional metadata.
{
    Ci = (color)texture (filename, u, v) * diffuse(N);   //< Beast expects the shader to set the output
                                                         //< closure in the Ci variable.
}
```

For a more complex and interesting example, see the `beastphong.osl` file, which models a Phong-type shading similar to the one used by Beast to render "classic" non-physical materials (see also Creating Classic Materials).

Setting up a physical material

Once you have your shader file ready, you need to:

1. Create a new `ILBMaterialHandle` .
2. Initialize the handle by calling `ILBCreateMaterial()` . In your call, you have to specify the handle of the scene that will contain your material, and the name of the material. This name should match the name of the material assigned to the triangles in your mesh.
3. Create a new `ILBShaderHandle` .
4. Initialize the shader by calling `ILBCreateShader()` . In your call, you have to specify the handle of the scene that will contain your shader, and the path to your shader file.
5. Assign your shader to your material by calling `ILBSetShader()` .
6. Set up any input parameters needed by the shader. See Creating Physical Materials below.

```
// Create and initialize the material.
ILBMaterialHandle material;
ILBCreateMaterial(scene, _T("materialWithShader"), &material);

// Create and initialize the shader.
ILBShaderHandle diffuseShader;
ILBCreateShader(scene, _T("diffuse"), _T("diffuseTexture.osl"), &diffuseShader);

// Assign the shader to the material.
ILBSetShader(material, diffuseShader);

// Create a new texture, and bind it to the material as an input parameter for the shader.
// Note that the call to ILBSetShaderParamTexture() specifies the name "diffuseFile", which is the
// name of the input parameter in the simple shader file shown above.
ILBTextureHandle tex;
ILBReferenceTexture(manager, _T("AUniqueName"), _T("C:\textures\wood.exr"), &tex);
ILBSetShaderParamTexture(material, _T("diffuseFile"), tex);
```

Note that this is a very simple example. A more in-depth usage might involve iterating over the attributes accepted by the shader, querying the type of each attribute (float, color, texture, etc.), and binding the necessary values to the material.

For a more complex example, look at the source for the Maya plugin. A Python script is used to extract the attributes accepted by the shader. These attributes are used to set up the BeastOSL node in Maya. The user can set values in the Maya UI. The settings in the Maya node are then queried, parsed, and bound to the material using that shader. See the `createMaterial()` method in `scenemanager.cpp`.

Binding input parameters

You can use the Beast API to set the following types of input parameters for your shaders:

- Floating-point numbers, using `ILBSetShaderParamFloat()`.
- Integers, using `ILBSetShaderParamInt()`.
- Textures, using `ILBSetShaderParamTexture()`.
- RGB colors, using `ILBSetShaderParamColor()`.
- UV sets, using `ILBSetShaderParamUV()`.

Each of these functions requires a material handle, a string that matches the name of the corresponding input parameter declared in the shader file, and the value that you want to pass to the shader.

Note that the input parameters are bound to a *material* handle, rather than to the *shader* handle. This allows you to use the same shader for multiple different kinds of materials, but to specify different input parameters for each of the different materials.

For example, in the `examples-physical` project, two different materials are created using the same shader, and set up with different parameters:

```
ILBShaderHandle phongishShader;
bex::apiCall(ILBCreateShader(bmh, _T("PhongishShader"), "../../data/phongish.osl", &phongishShader));
...

// Sphere 1 - Textured Lambert
ILBMaterialHandle sm1;
bex::apiCall(ILBCreateMaterial(scene, _T("TexturedLambert"), &sm1));
bex::apiCall(ILBSetShader(sm1, phongishShader));
bex::apiCall(ILBSetShaderParamTexture(sm1, _T("DiffuseTexture"), xorTexture));
bex::apiCall(ILBSetShaderParamColor(sm1, _T("DiffuseColor"), &ILBLinearRGB(0.5f, 0.5f, 1.0f)));
bex::apiCall(ILBSetMaterialOverrides(sphereInstances[1], &sm1, 1));

// Sphere 2 - Phongish
ILBMaterialHandle sm2;
bex::apiCall(ILBCreateMaterial(scene, _T("Phongish"), &sm2));
bex::apiCall(ILBSetShader(sm2, phongishShader));
bex::apiCall(ILBSetShaderParamColor(sm2, _T("DiffuseColor"), &ILBLinearRGB(0.0f, 0.0f, 0.0f)));
bex::apiCall(ILBSetShaderParamColor(sm2, _T("SpecularColor"), &ILBLinearRGB(1.0f, 0.0f, 0.0f)));
bex::apiCall(ILBSetShaderParamColor(sm2, _T("Shininess"), &ILBLinearRGB(400.0f,400.0f,400.0f)));
bex::apiCall(ILBSetMaterialOverrides(sphereInstances[2], &sm2, 1));
```

Despite using the same shader, the result of rendering the two spheres is very different.

Thread safety

You can create multiple different materials simultaneously in multiple threads. In addition, multiple threads can find materials in the cache and use them simultaneously. You can also modify a single material from multiple different threads at the same time.

Related API functions

API functions related to the creation and setup of materials are declared in the `beastmaterial.h` file.

Topics in this section

EXHIBIT 135

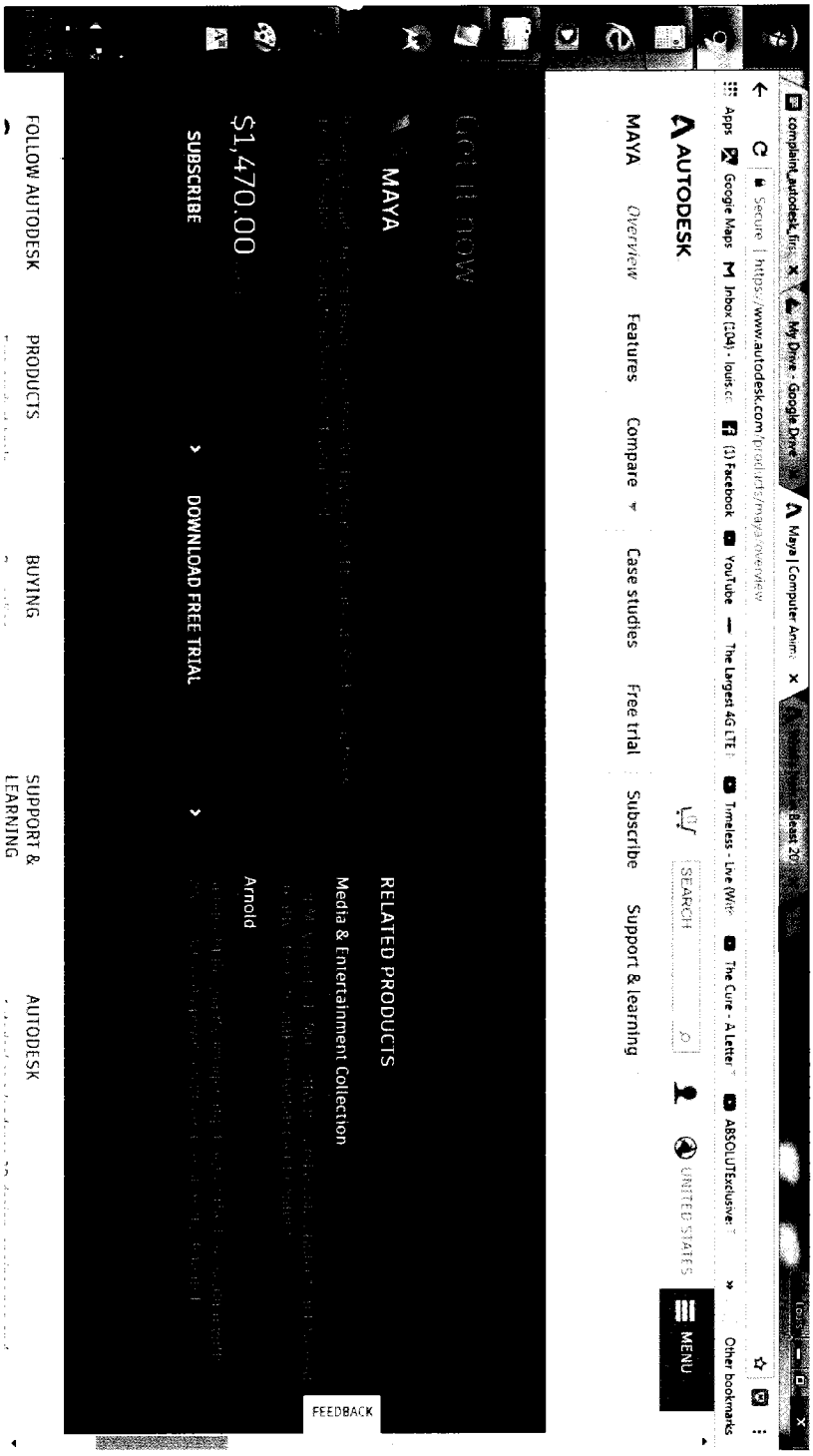

AUTODESK
MAYA
Overview
Features
Compare
Case studies
Free trial
Subscribe
Support & learning
SEARCH
MENU
MAYA
RELATED PRODUCTS
Media & Entertainment Collection
Arnold
$1,470.00
SUBSCRIBE
DOWNLOAD FREE TRIAL
FEEDBACK
FOLLOW AUTODESK
PRODUCTS
BUYING
SUPPORT & LEARNING
AUTODESK

EXHIBIT 136

COFFELT

AUTODESK

EXHIBIT 137

COFFEE CUP

AUTODESK

EXHIBIT 138

COFFELT

AUTODESK

EXHIBIT 139

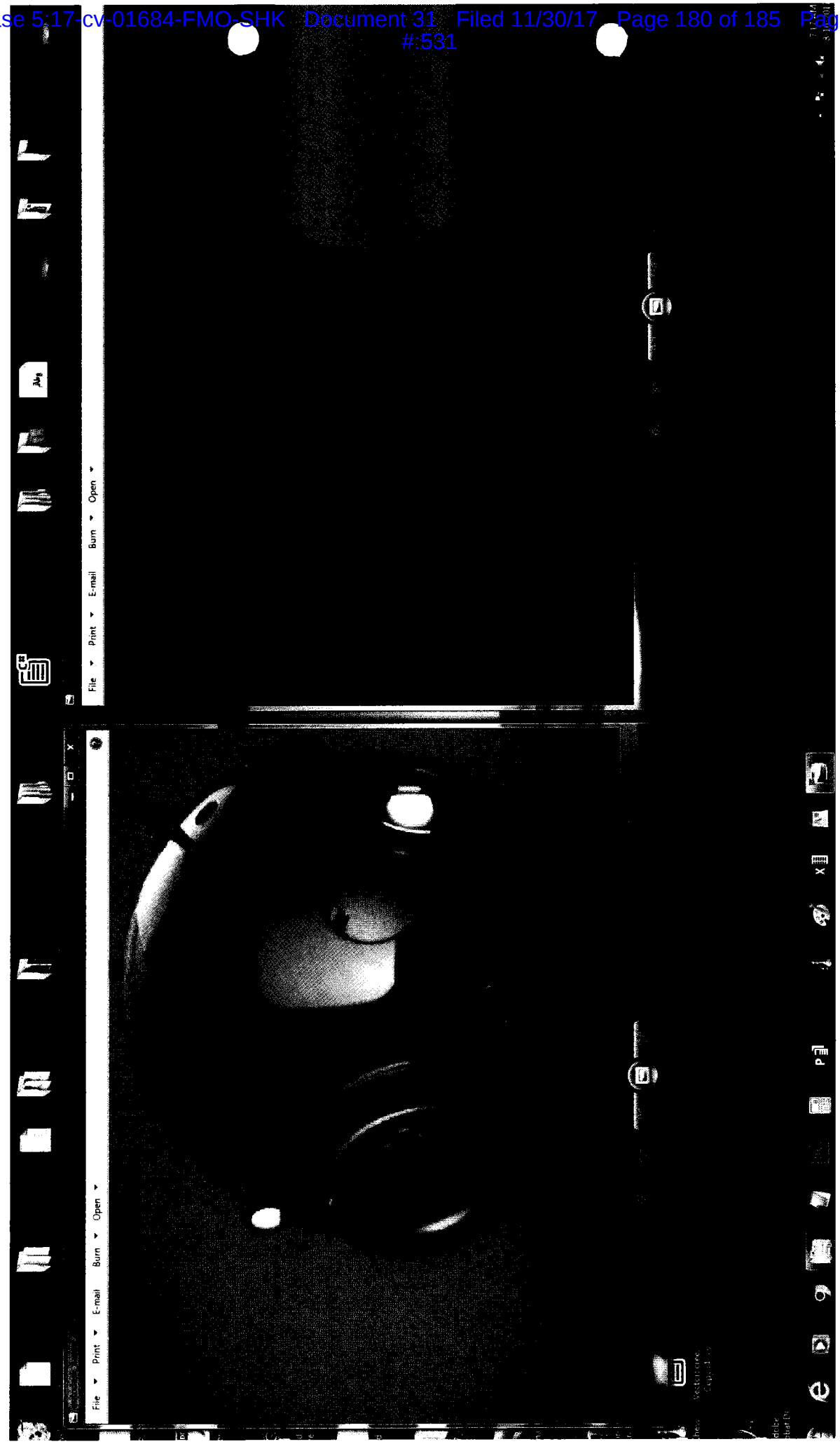
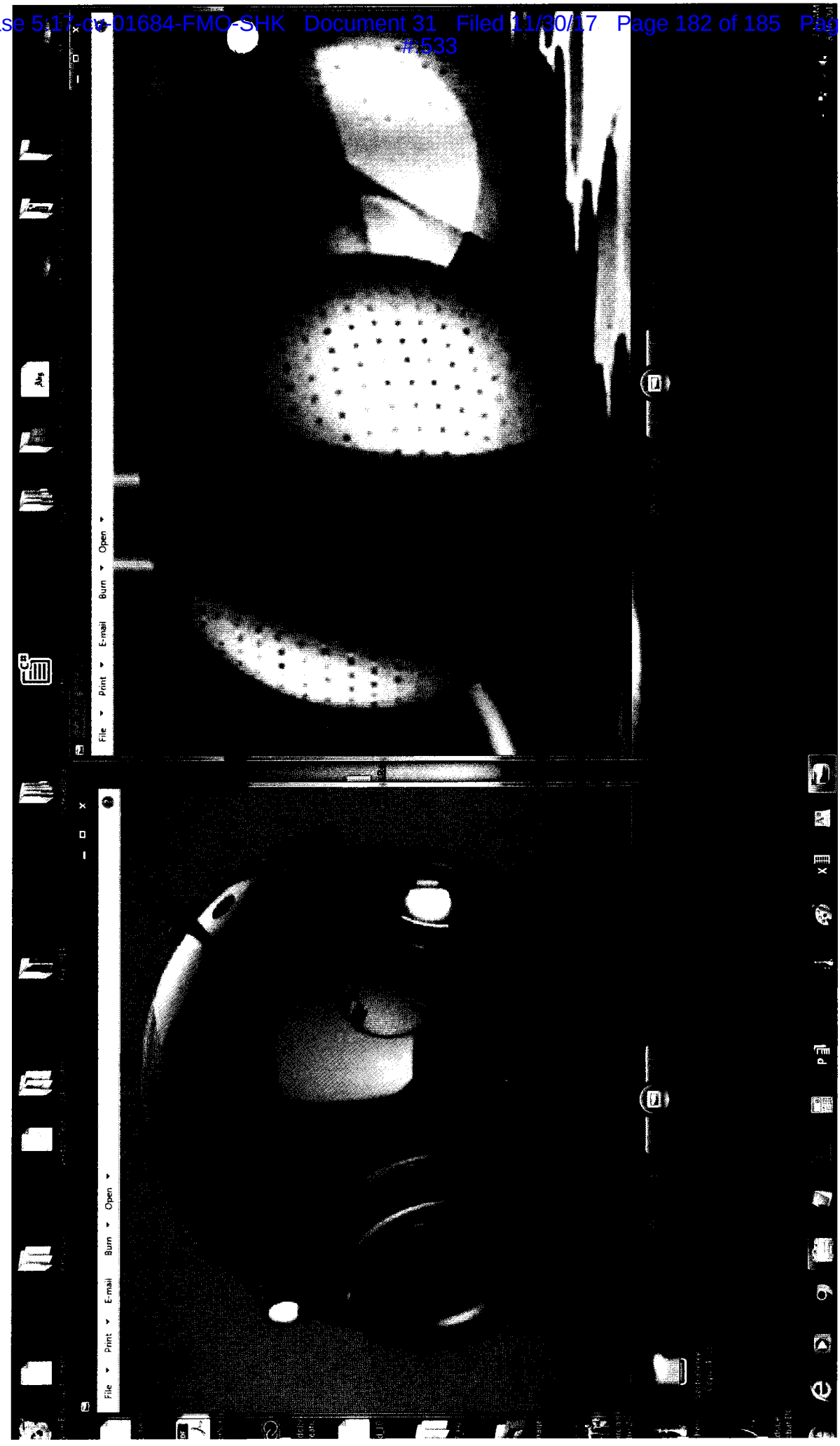

File
Print
E-mail
Burn
Open
COFFEE
AUTODESK

EXHIBIT 140



COFFELT

AUTODESK

EXHIBIT 141

File
Print
E-mail
Burn
Open
COFFELT
AUTODESK

CERTIFICATE OF SERVICE

I, Louis A. Coffelt, Jr. ("Coffelt") hereby certify that on the 30th day of November, 2017, Coffelt filed the the foregoing document **FIRST AMENDED COMPLAINT FOR COPYRIGHT INFRINGEMENT** with the Clerk of the Court, in case No. 5:17-cv-01684-FMO-SHK, as follows:

Office of the Clerk
United States District Court for the Central District of California
350 W. 1st Street, Los Angeles, CA 90012

And further caused to be served one copy of the foregoing document **FIRST AMENDED COMPLAINT FOR COPYRIGHT INFRINGEMENT** by U.S. mail, postage prepaid, in case 5:17-cv-01684-FMO-SHK to the following attorneys of record:

RICHARD S.J. HUNG
rhung@mofo.com
Morrison & Foerster LLP
425 Market Street
San Francisco, CA 94105
(415) 268-7602 (direct)
(415) 268-7522 (fax)

Date: November 30, 2017

By: Louis A. Coffelt, Jr.

Louis A. Coffelt, Jr.

Plaintiff

Pro Se